Intel® Iris® Plus Graphics and UHD Graphics Open Source

Programmer's Reference Manual

For the 2019 10th Generation Intel Core™ Processors
based on the "Ice Lake" Platform

Volume 6: Memory Views

January 2020, Revision 1.0

Table of Contents

Memory Views 1

Introduction 1

Memory Views Glossary 1

Graphics Memory Address Spaces 1

Graphics Virtual Memory 3

Graphics Translation Tables 3

GFX Page Tables 4

Tiled Resources Translation Tables 4

Registers for TR-TT Management 6

Detection and Treatment of Null and Invalid Tiles 9

TR-TT Modes 10

Virtual Addressed TR Translation Tables 10

TR-TT Page Walk 11

Gen9 Page Table Modes 12

Gen9 Per Process GTT 12

Page Tables Entry (PTE) Formats 12

Pointer to PML4 Table 17

PML4E: Pointer to PDP Table 17

PDPE: Pointer to PD Table 19

PD: Pointer to Page Table 22

PTE: Page Table Entry for 64KB Page 24

PTE: Page Table Entry for 4KB Page 26

PPGTT for 32b Virtual Address 27

Walk with 64KB Page 30

Walk with 2MB Page 31

Walk with 1GB Page 31

PPGTT for Standard Context (64b VA) 31

Walk with 64KB Page 39

Walk with 2MB Page 40

Walk with 1GB Page 41

Gen9 Global GTT 42

Page Table Entry 42

Page Walk 42

Legacy mode with 32b VA 43

Page Walk in Legacy mode with 32b VA 43

Walk with 64KB Page 45

Page Table Entry (PTE) Formats 45

PDE for Page Table 46

PTE: Page Table Entry for 64KB Page 47

PTE: Page Table Entry for 4KB Page 48

Legacy mode with 48b VA 48

Page Walk in Legacy 48b Mode 49

Walk with 64KB Page 50

Walk with 2MB Page 51

Walk with 1GB Page 52

Page Tables Entry PTE Formats 52

Pointer to PML4 table 53

PML4E: Pointer to PDP Table 54

PDPE: Pointer to PD Table 54

PDPE for PD 55

PDPE for 1GB Page 56

PD: Pointer to Page Table 57

PDE for Page Table 57

PDE for 2MB Page 58

PTE: Page Table Entry for 64KB Page 59

PTE: Page Table Entry for 4KB Page 61

Advanced mode with 48b VA and IA32e Support 61

Page Walk in Advanced Mode 62

Walk with 2MB Page 63

Walk with 1GB Page 64

Page Tables Entry (PTE) Formats 64

Pointer to PML4 table 65

PML4E: Pointer to PDP Table 66

PDPE: Pointer to PD Table 67

PDPE for PD 67

PDPE for 1GB Page 69

PD: Pointer to Page Table 70

PDE for Page Table 70

PDE for 2MB Page 72

PTE: Page Table Entry for 64KB Page 73

PTE: Page Table Entry for 4KB Page 75

GTT Cache 76

GFX Page Walker (GAM) 76

Context Definition for GFX Page Walker 77

Context Definition Delivery 78

Element Descriptor Register 79

PDP0/PML4/PASID Descriptor Register 80

PDP1 Descriptor Register 81

PDP2 Descriptor Register 81

PDP3 Descriptor Register 82

List of Registers and Command Streamers 82

Updating Page Table Pointers (aka PD Load) 84

Page Walker (GAM) Reset 85

TLB Caching and Management 85

TLB Caches 87

Context Cache - CC 87

PASID Cache - PC 87

Intermediate Page Walk Caches (PML4, PDP, PD) – PWC 87

TLB – Final Page Entry 87

TLB Entry Content 88

TLB Accessed and Dirty Flags 90

Updating A/D Bits 91

Replacement 96

Invalidations of TLB 96

Optional Invalidations 96

Atomic Operations between GPU and IA 97

Memory Views

Introduction

The hardware supports three engines:

- The Render command streamer interfaces to 3D/IE and display streams.
- The Media command streamer interfaces to the fixed function media.
- The Blitter command streamer interfaces to the blit commands.

Software interfaces of all three engines are very similar and should only differ on engine-specific functionality.

Memory Views Glossary

Term	Definition
IOMMU	I/O Memory Mapping unit
SVM	Shared Virtual Memory, implies the same virtual memory view between the IA cores and processor graphics.
Page Walker (GAM)	GFX page walker which handles page level translations between GFX virtual memory to physical memory domain.

Graphics Memory Address Spaces

The *Graphics Memory Address Spaces* table lists the five supported Graphics Memory Address Spaces. Note that the Graphics Memory Range Removal function is automatically performed to transform system addresses to internal, zero-based Graphics Addresses.

Graphics Memory Address Types

Address Type	Description	Range	Gen9
GMADR	Address range allocated via the Device 2 (integrated graphics device) GMADR register. The processor and other peer (DMI) devices utilize this address space to read/write graphics data that resides in Main Memory. This address is internally converted to a GM_Address.	This is a 4 GB bar above physical memory.	128 MB, 256 MB, 512 MB, 1GB, 2GB, 4GB
GTTMMADR	The combined Graphics Translation Table Modification Range and Memory Mapped Range. The range requires 16 MB combined for MMIO and Global GTT aperture, with 8MB of that used by MMIO and 8MB used by GTT. GTTADR will begin at GTTMMADR 8MB while the MMIO base address will be the same as GTTMMADR. For the Global GTT, this range is defined as a memory bar in graphics device config space. It is an alias into which software is required to write Page Table Entry values PTEs.	This is a 16MB bar above physical memory.	16 MB (2 MB MMIO + 6 MB reserved + 8 MB GGTT)

Address Type	Description	Range	Gen9
	Software may read PTE values from the global Graphics Translation Table GTT. PTEs cannot be written directly into the global GTT memory area. GTTMMADR must be marked uncache-able (UC). Accesses to GTTMMADR(GTT) must be naturally aligned and 64 bits or less (ie. 1 GTT entry). Accesses to GTTMMADR(Register) must be naturally aligned and 32 bit or less.		
GSM	GTT Stolen Memory. It is an 8 MB (max) region taken out of physical memory to store the Global GTT entries for page translations specific to GFX driver use. It is accessible via GTTMMADR from the CPU path however GPU/DE can access the same region directly.	This is an 8 MB region in physical memory not visible to OS.	1 MB, 2 MB, 4 MB, 8 MB
DSM	Data stolen memory, the size is determined with GMS filed (8 bits) with MAX size of 4 GB. This is a stolen memory which can be accessed via GMADR for CPU and directly for GPU/DE. Size is programmable with 32 MB multiplier. First 4KB of DSM has to be reserved for GFX hardware use.	This is a max of 4 GB stolen physical memory for GFX data structures.	0 MB, 32 MB, 64 MB, 96 MB, ...4096MB
PCM/WOPCM	Reserved within the DSM for protected content functions.	Limited by the DWM size, base is programmable.	

Next level breakdown for GTTMMADR is given below.

Software is allowed to use range x17_0000 to x17_FFFF as the Null range.

Graphics Virtual Memory

The GPU uses a virtual memory address space, where the graphics virtual address is mapped through a Page Table (PPGTT) to a physical memory address. Normally, this mapping is set up by the graphics device driver and is private to the GPU context. However, in some cases the graphics virtual address is shared with the CPU – see Shared Virtual Memory (SVM) for more information.

The range of valid graphics virtual addresses, and the types of page tables supported for address translation, varies with the GPU configuration. See the Configurations section for a summary the ranges and features supported by a specific graphics device.

Although the range of supported graphics virtual addresses varies, most GPU commands and GPU instructions use a common 64-bit definition for a graphics virtual address. Addresses outside of the supported range are reserved for future address space expansion. See the **GraphicsAddress** structure definition for specific details.

Some GPU devices support an extended graphics virtual memory address mapping called Tiled Resources. When enabled, the Tiled Resources Translation Table (TR-TT) pre-processes graphics virtual addresses. TR-TT maps a graphics virtual memory address either to a new graphics virtual memory address or to a Null Tile. Null Tiles return zero on reads and drop writes. For translations that are not Null Tiles, the new graphics virtual memory address is then used for the graphics virtual address and translated through the normal Page Table to generate a physical memory address.

Graphics Translation Tables

GT supports standard virtual memory models as defined by the IA programmer's guide. This section describes the different paging models, their behaviors, and the page table formats.

The Graphics Translation Tables GTT (Graphics Translation Table, sometimes known as the global GTT) and PPGTT (Per-Process Graphics Translation Table) are memory-resident page tables containing an array of DWord Page Translation Entries (PTEs) used in mapping logical Graphics Memory addresses to physical memory addresses, and sometimes snooped system memory "PCI" addresses.

The base address (MM offset) of the GTT and the PPGTT are programmed via the PGTBL_CTL and PGTBL_CTL2 MI registers, respectively. The translation table base addresses must be 4KB aligned. The GTT size can be either 128KB, 256KB, or 512KB (mapping to 128MB, 256MB, and 512MB aperture sizes respectively) and is physically contiguous. The global GTT should only be programmed via the range defined by GTTMMADR. The PPGTT is programmed directly in memory. The per-process GTT (PPGTT) size is controlled by the PGTBL_CTL2 register. The PPGTT can, in addition to the above sizes, also be 64KB in size (corresponding to a 64MB aperture). Refer to the GTT Range chapter for a bit definition of the PTE entries.

GFX Page Tables

GPU supports three-page table mechanisms

- IA32e compatible GTT
- PPGTT – private per process GTT (private GFX)
- GGTT - global GTT

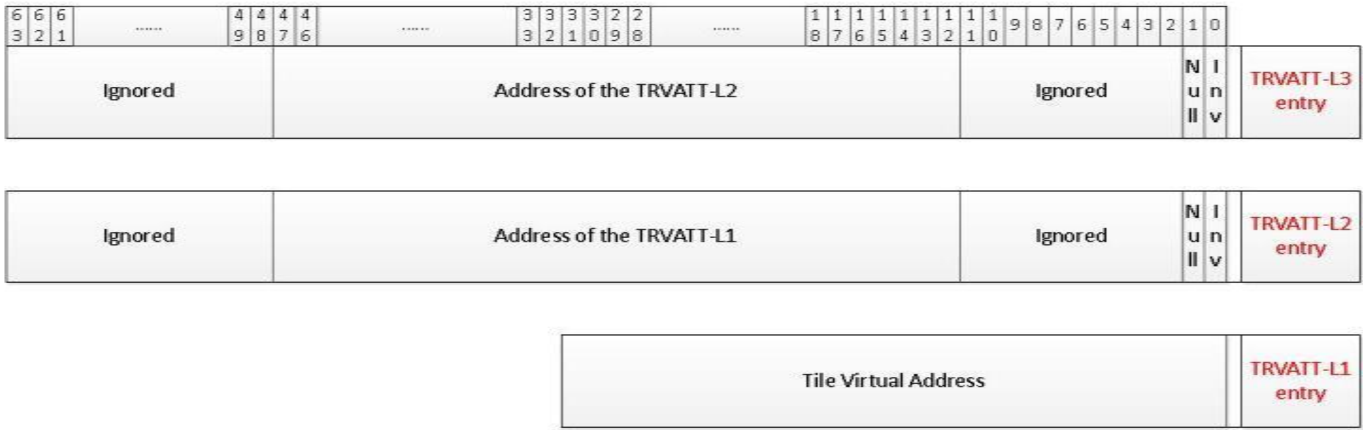
All page tables use the same 64-bit PTE format. Differences are in how various bit fields applies (vs reserved) under various usage models.

Gen9 follows the same principles that gen8 set it up for improved page tables and compatibility of OS managed page table formats.

Tiled Resources Translation Tables

Sparse Tiled Resources can be thought of as a kind of application-controlled virtual memory scheme. The application allocates a resource in a virtual address space. Then the application tells the driver to map specified 64KB tiles within the surface to memory, within resources called Tile Pools. Tiles that are not mapped to a Tile Pool are null tiles.

Tiled Resource Translation Table (TRTT) is constructed as a 3-level tile Table. Each tile is 64KB in size which leaves behind 44-16=28 address bits. 28bits are partitioned as 9+9+10 which corresponds to TRVATT L3, L2 and L1 respectively. This is where TRVATT L3 has 512 entries, L2 has 512 entries and L1 has 1024 entries where each level is contained within a 4KB page hence L3 and L2 is composed of 64b entries and L1 is composed of 32b entries.



The contents of the TRVATT tables are as listed above where L3 and L2 points to the address of the next level which is a 4KB page and L1 contains the 32b VA address pointer needed to map the TR tile to virtual address space.

L1 Entry:

Bits	Field	Description
31:0	ADDR: Address	GFX virtual address of 64KB tile is referenced by this entry. This field is treated as GFX Virtual Address (GVA) when translated and maps to 47:16.

L2 Entry:

Bits	Field	Description
63:48	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L1. *TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.*
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	Null	Null Tile where reads to this tile returns zero with a Null indicator and writes are dropped.
0	Invalid	Invalid Tile where reads to this tile returns zero and writes are dropped. Additional interrupt is generated to GFX software when an invalid tile is accessed.

L3 Entry:

Bits	Field	Description
63:48	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L2. *TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.*
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	Null	Null Tile where reads to this tile returns zero with a Null indicator and writes are dropped.
0	Invalid	Invalid Tile where reads to this tile returns zero and writes are dropped. Additional interrupt is generated to GFX software when an invalid tile is accessed.

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
GFX Driver has to disable the TR-TT bypass mode before using tiled resources translation tables. Details of the registers are given in "registers for TR-TT management."	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
GFX Driver is not allowed to put TR-TT entries into TR-VA space.	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
Usage model for TR translations are restricted to GFX Render Engine (& POSH pipeline).	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
TRTT is only for PPGTT64 (Advanced or Legacy PPGTT64). Enabling TRTT in Legacy PPGTT32 context or GGTT context is considered as invalid programming.	

Registers for TR-TT Management

Following register is a global mechanism to disable the bypass mode which is considered to be default for h/w. GFX driver has to set this bit to disable bypass mode before using TR-TTs.

Following registers shall be part of the h/w context.

Tiled Resources VA Translation Table L3 Pointer

Register Space:	MMIO: 0/2/0

DWord	Bit	Description
1	63:48	**Reserved** **Access:** RO Reserved.
	47:32	**Tiled Resource – VA translation Table L3 Pointer (Upper Address)** **Default Value:** 0000h **Access:** R/W Upper address bits for tiled resource VA to virtual address translation L3 table. For physical memory option, address bits [47:39] has to be programmed to "0" as it is defined the limit of physical memory allocation.
0	31:16	**Tiled Resource – VA translation Table L3 Pointer (Lower Address)** **Default Value:** 0000h **Access:** R/W Lower address bits for tiled resource VA to virtual address translation L3 table.
	15:0	**Reserved** **Access:** RO Reserved.

Tiled Resources Null Tile Detection Register

Register Space:	MMIO: 0/2/0

DWord	Bit	Description
	31:0	**Null Tile Detection Value** **Default Value:** 00000000h **Access:** R/W A 32bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Null Tiles. Hardware will flag each entry and space behind it as Null Tile for matched entries.

Tiled Resources Invalid Tile Detection Register

Register Space:	MMIO: 0/2/0

DWord	Bit	Description
	31:0	**Invalid Tile Detection Value** **Default Value:** 00000000h **Access:** R/W A 32bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Invalid Tiles. Hardware will flag each entry and space behind it as Invalid Tile for matched entries.

Tiled Resources Virtual Address Detection Registers (TRVADR)

Register Space:	MMIO: 0/2/0

DWord	Bit	Description
0	31:8	**Reserved** **Access:** RO Reserved.
	7:4	**TRVA Mask Value (TRVAMV)** **Default Value:** 0000b **Access:** R/W 4bit MASK value that is mapped to incoming address bits[47:44]. MASK bits are used to identify which address bits need to be considered for compare. If particular mask bit is "1", mapping address bit needs to be compared to DATA value provided. If "0", corresponding address bit is masked which makes it don't care for compare (*this field defaults to "0000" to disable detection*) *Note that h/w supports two possible values for MASK: "0000" which is disabled case and "1111" where*

Tiled Resources Virtual Address Detection Registers (TRVADR)		
		44 bit TR-VA space is carved out.
	3:0	**TRVA Data Value (TRVADV)** **Default Value:** 0b **Access:** R/W 4bit DATA value that is mapped to incoming address bits[47:44]. Data bits are used to compare address values that are not filtered by the TRVAMV for match.

Tiled Resources Translation Table Control Register (TRTTE)		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
0	31:2	**Reserved** **Access:** RO Reserved.
	1	**TR-VA Translation Table Memory Location** **Default Value:** 0b **Access:** R/W This fields specifies whether the translation tables for TR-VA to VA are in virtual address space vs physical (GPA) address space. 0: Tables are in Physical (GPA) Space 1: Tables are in Virtual Address Space **Tiled Resource Translation Tables in GPA space is not supported in any GEN generations. For Gen9/10/11, this mode should never be set as GPA mode (always set to '1).**
	0	**TR-TT Enable** **Default Value:** 0b **Access:** R/W TR translation tables are disabled as default. This field needs to be enabled via s/w to get TR translation active.

The following register (0x4DFC[0]) has enable and disable control of the bypass path across TR translations. This control exists in Gen9/10. By default, bypass is enabled, and bypass needs to be disabled (by setting 0x4DFC[0] = '1) for TR translations to function. Disabling the bypass should be done before render power gating is enabled.

Detection and Treatment of Null and Invalid Tiles

Two types of definition that need to be extracted from TR-VA walk in addition to reaching the GFX virtual address.

1. **Null Tiles**: Null tiles provide the applications the of capability to preventing OS mapping the entire surface. When a memory access hits a Null tile, the access is terminated and zero's are returned to the originator of the memory access for loads along with a null indicator and for stores the access is dropped at the page walker level.
2. **Invalid Tiles**: This is the case where GFX software did not update the value of the mapping properly for hardware to separate resident vs null tiles. The Invalid Tile treatment is exactly same however additionally a unique interrupt is generated in h/w.

Both detections are done by GPU:

- For L2/L3 entries, Null and Invalid tile information is already embedded in the TR-TT entries
- For L1 entries, the contents (32bits) are compared in hardware to pre-programmed values by GFX software (*values are provided in GFX MMIO space*). For the match values, two separate 32b registers are defined, one for Null Tile detection and one for Invalid Tile detection.

Hardware walking matching the value or detecting L2/L3 would terminate the walk (i.e. rest of the tables are not valid) and define the access as either Null or Invalid.

Programming Note	
Context:	Detection and treatment of null and invalid tiles.
The software is not allowed to program both Null and Invalid values to be the same.	

Programming Note	
Context:	TileX Surfaces and Null Tiles
NULL or Invalid Tiles are not supported on TileX surfaces.	

GPU implements a counter mechanism to roll-up the Null tile accesses detected. The counter value is exposed to GFX software via GFX MMIO.

In Gen9/Gen10 implementation, when the TR translation tables are in Gfx virtual address domain, the pages faults encountered while walking the IA32e pages are not reported back to the TR walkers or TLBs. These faults are handled as fault & halt, making these faults transparent to the TR walkers. However, when such a fault is not fixed (unsuccessful fault response) or when a non-recoverable fault encountered, main page walker HW converts the cycle to an invalid cycle. Thus, in this case, TR walker or TR TLBs will get incorrect read return data without any notification of the non-recoverable fault condition. Thus, TR walker/TLBs will continue with the TR-walk with incorrect data. This can lead to spurious cycles being generated. However, a Gfx reset/FLR is expected as a result of the non-recoverable fault.

TR-TT Modes

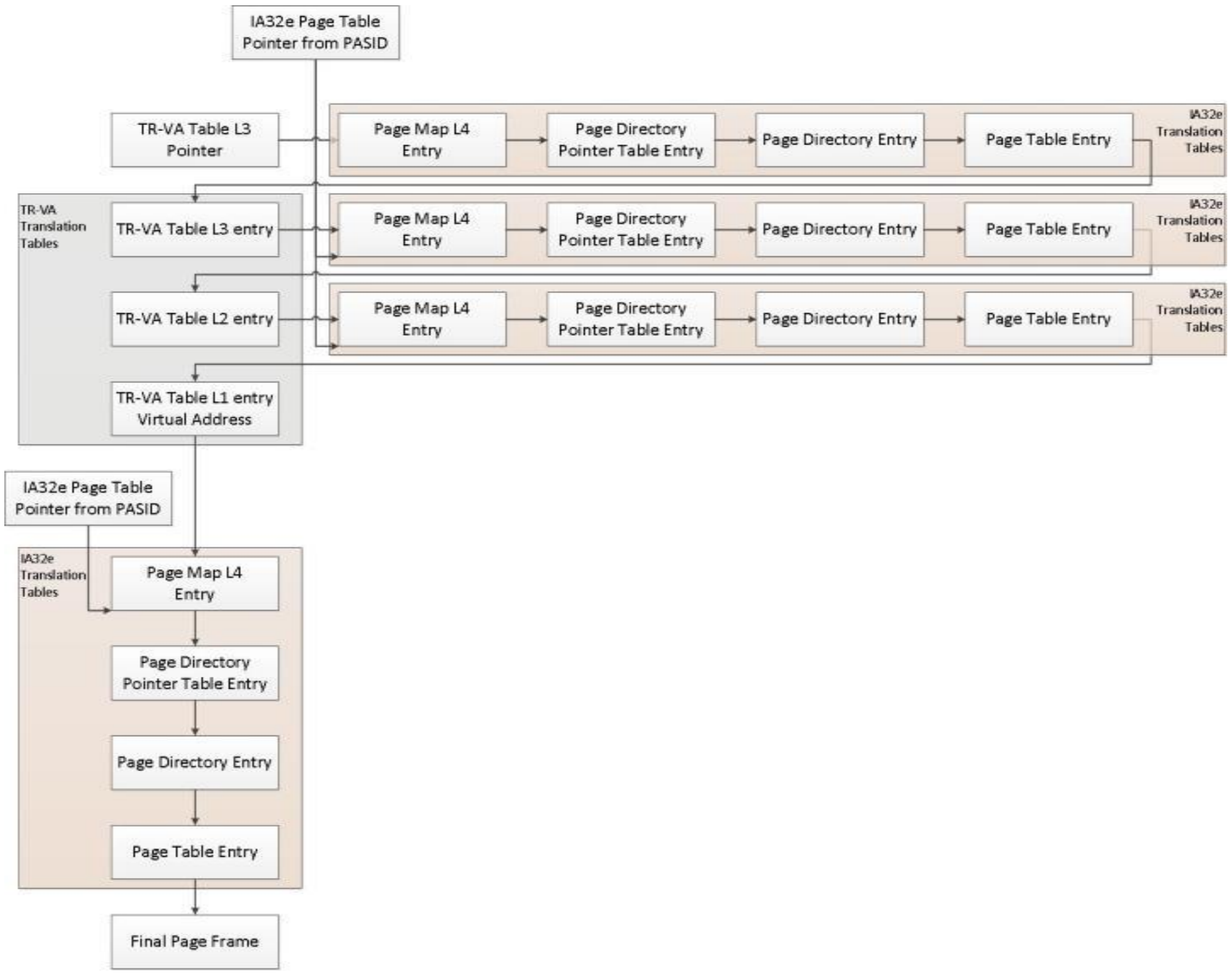
The L3 table pointer along with TRTTL3e/TRTTL2e is projected to support two modes of address space. Original intent was to have the contents to be in Virtual Address space (OS managed) and have them to be translated to GPA to HPA before getting accessed. Such mechanism will incur high latency penalties due to nested page translations. GPU shall have an additional mode where tiled-resources translation tables are in physical address space (GPA) and eliminate the need to have nested translations to reduce the potentially high miss latencies.

TR-TT walker shall have both modes supported. The Mode bit will be part of the same Register that provides TR-VA TT L3 pointer.

Virtual Addressed TR Translation Tables

Having sparse tiled resource translation tables in GFX virtual space requires the h/w TR-TT walker to walk thru the 1st level tile tables for table accesses to reach to Physical address at the L1 TR translation tables.

The following diagrams provide the view of the walk TR-VA translation tables are in physical memory and no 2nd Level (VTd) translations enabled.



Once 2nd level translations are enabled each level of 1st level walk needs to be further walked through VTd page tables.

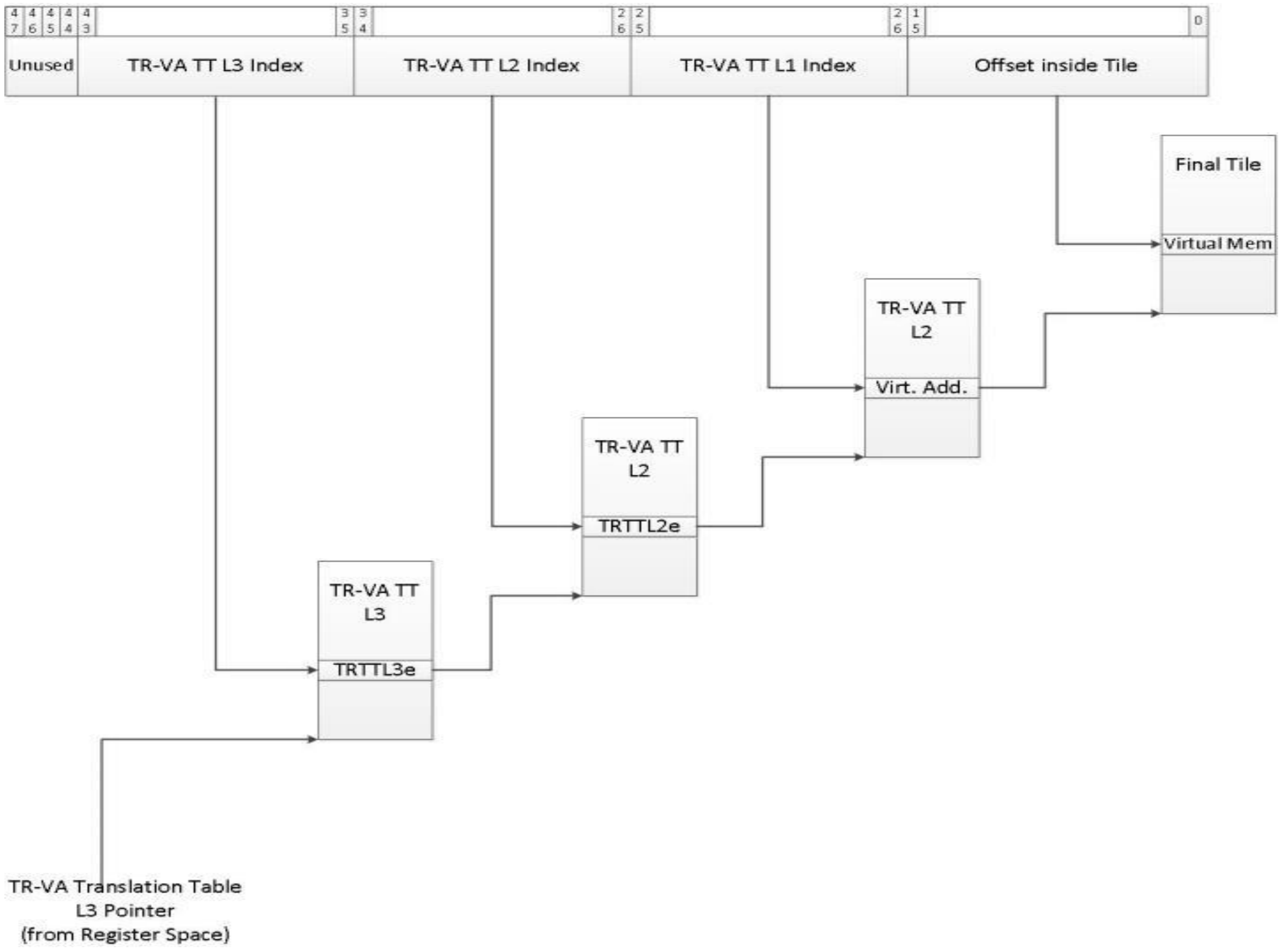
The level of nested walks does not change the structure of the TR-VA walker; it just defines the recursive nature of the translations.

TR-TT Page Walk

Sparse Tiled Resources translation tables are separated into 3-levels. The pointer to L3 table is going to be set up in GFX MMIO space as part of the context, this pointer be would be available to page walker ahead of any TR-VA memory accesses.

TR-TT L3 walk will be consistent of calculating the 64b of interest based on the L3 table pointer and using the 9 bit index (address bits[43:35]). L2 will use TR-TT L3 entry as the table pointer and use the next set of 9 address bits ([34:26]) to locate the L2 entry which is a pointer to L1 table. Final L1 table is located with L2 entry and indexed by remaining 10 address bits (25:16) to index where 32b virtual address is extracted.

Post TR-TT walk 32b entry from L1 is mapped to final virtual address 47:16 and remaining 15:0 is passed from the original TR-VA access as is given all tiles in TR-VA space are 64KB in size.



Gen9 Page Table Modes

GFX Aperture and Display accesses are mapped thru Global GTT to keep the walk simple (i.e. 1-level) and latency sensitive. GPU accesses to memory can be mapped via Global GTT and/or ppGTT with various addressing modes.

Supported walk modes are listed as following:

1. **Global GTT with 32b virtual addressing**: Global GTT usage is similar to previous generations with extended capability of increasing virtual address (VA) up to 4GB (from 2GB) and use a standard 64b PTE format. The breakdown of the PTE for global GTT is given in later sections and allows 1-level page walk where the 20b index is used to select the 64b PTE from memory.
2. **Legacy 32b VA with ppGTT**: This is a mode where ppGTT page tables are considered private and managed via GFX sotfware (driver) where context is tagged as Legacy 32b VA. Each page walk is managed via 9b of the virtual address and 20b index to address 4GB memory space is broken into 3 parts. In order to optimize the walks and make it look like previous generations, GFX sotfware provides 4 pointers to page tables (called 4 PDP entries) all guest physical address. GPU uses the four pointers and fetches the 4x4KB into h/w (for render and media) before the context execution starts. The optimization limits the dynamic (on demand) page walks to 1-level only.
3. **Legacy 48b VA with ppGTT**: GFX address expansion beyond 4GB is added to address 48b virtual address space. 48b VA requires 36b indexing (4x9b) translating into 4-levels of page walk. To reduce the overhead of 4 level walk, GPU will cache the entire content of PML4 (4kB) to limit the on-demand walks to 3 levels. The caching happens as part of the initial demand where no further replacements required.
4. **Advanced 48b VA with IA32e support via IOMMU**: 48b addressing in advanced mode is managed via IOMMU settings where the base of the page table shall be found after the root / context tables using bus/device/function values. PASID# is used as an index in PASID table to find page table pointer to start the 4-level page walk. Rest of the mechanism is similar to Legacy 48b VA mode, GPU has the capability to cache entire content of PML4 and try to limit the dynamic page walks to 3-level.

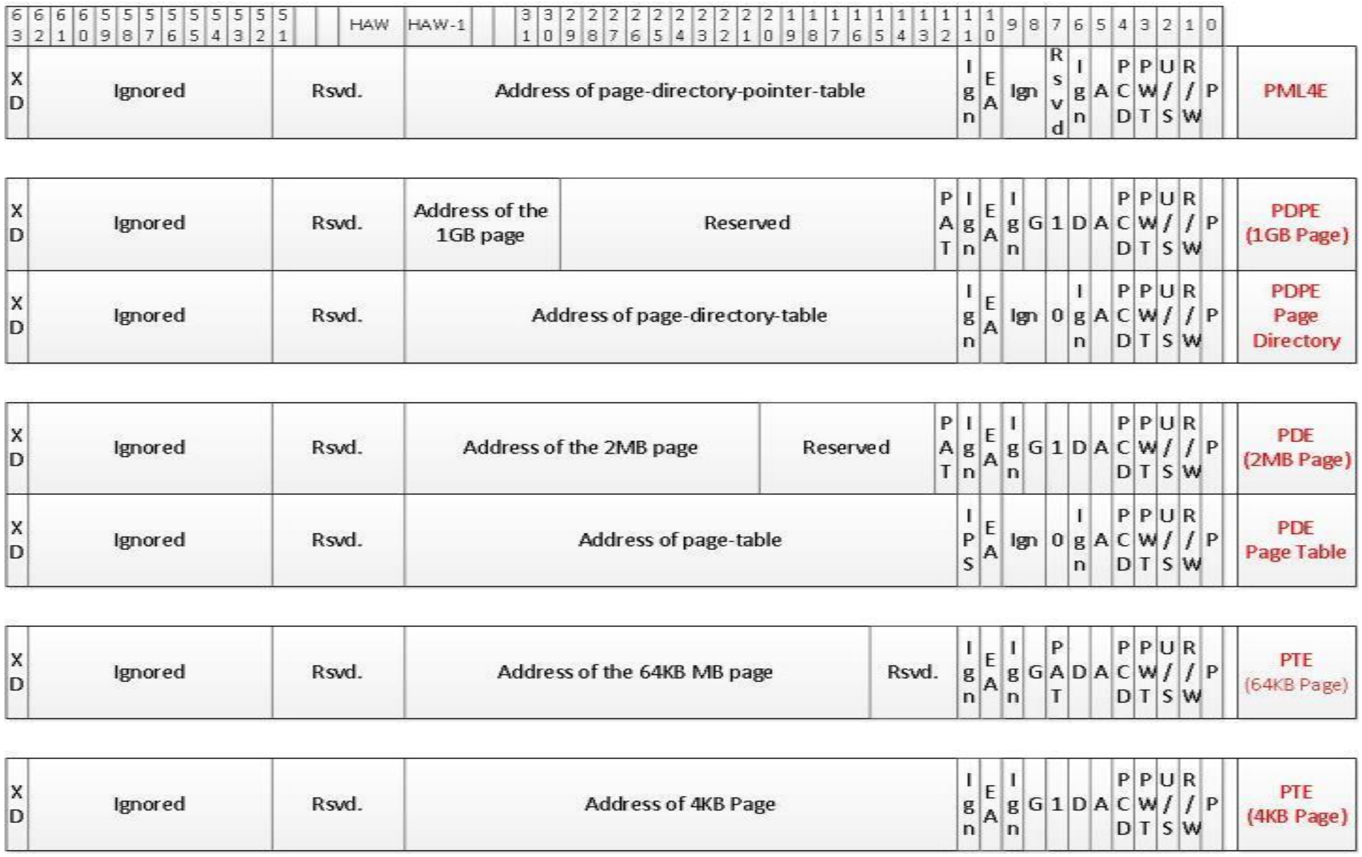
Gen9 Per Process GTT

Gen8/9 per process GTT mechanism has multiple hooks and mechanisms for s/w to prepare the page walks on hardware. The listed mechanisms here are selectable per-context and descriptors are delivered to hardware as part of context descriptor.

The entry contents are also modified to match the same format as IA32e page tables allowing future expansion for sharable page tables as well as higher order virtual addressing.

Page Tables Entry (PTE) Formats

Page Table Entry (PTE) formats follow the IA32e layout shown below. Note that the Hardware Address Width (HAW) is determined by Uncore: typically, 39 for client products and 46 for server products.

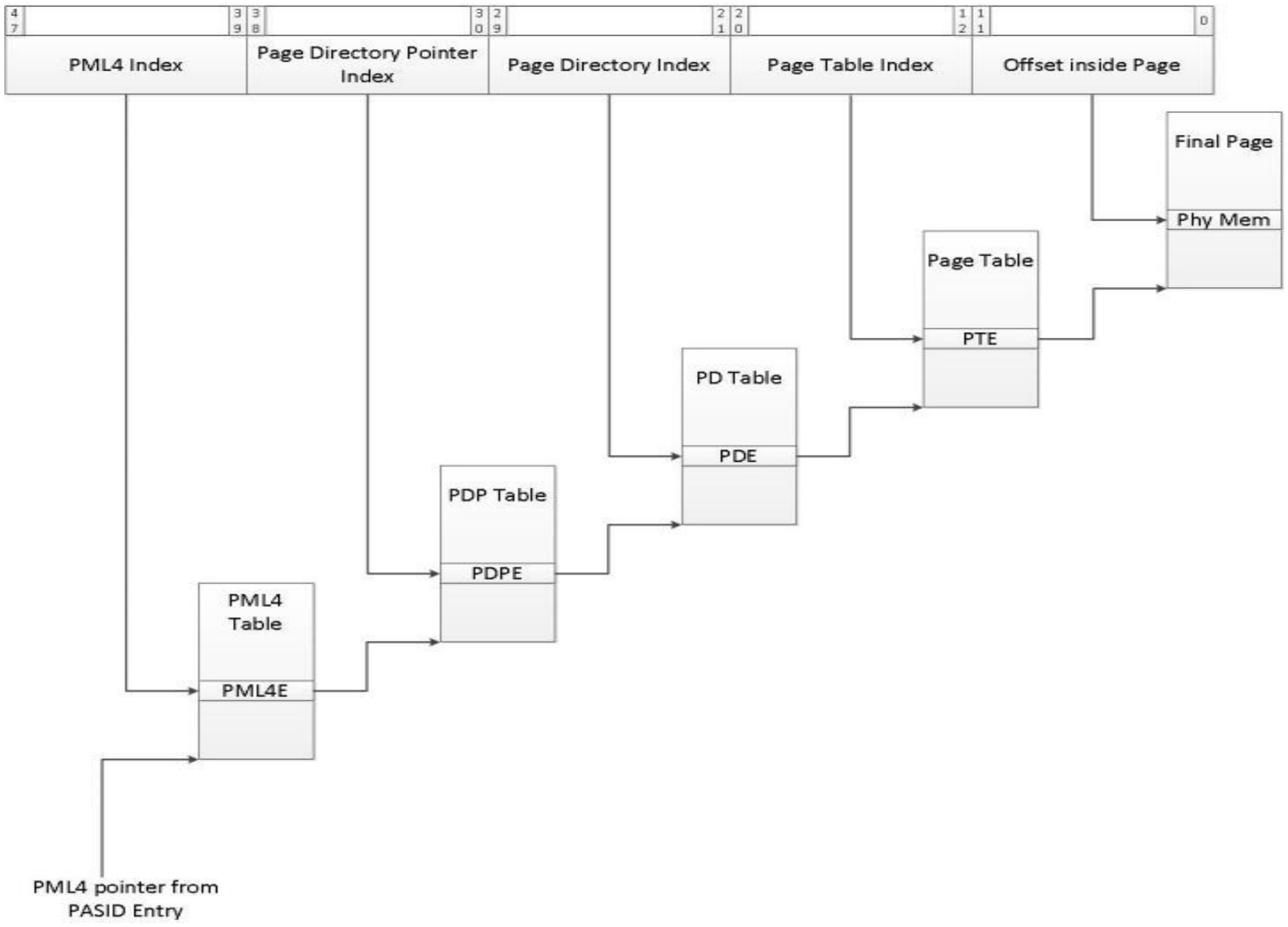


63	62 61 60 59 58 57 56 55 54 53 52	51	HAW HAW-1 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13		12	11	10	9	8	7	6	5	4	3	2	1	0	
XD	Ignored	Rsvd.	Address of page-directory-pointer-table			Ign	EA	Ign		Rsvd	Ign	A	PCD	PWT	U/S	R/W	P	PML4E
XD	Ignored	Rsvd.	Address of the 1GB page	Reserved	PAT	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P	PDPE (1GB Page)
XD	Ignored	Rsvd.	Address of page-directory-table			Ign	EA	Ign		0	Ign	A	PCD	PWT	U/S	R/W	P	PDPE Page Directory
XD	Ignored	Rsvd.	Address of the 2MB page	Reserved	PAT	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P	PDE (2MB Page)
XD	Ignored	Rsvd.	Address of page-table			IPS	EA	Ign		0	Ign	A	PCD	PWT	U/S	R/W	P	PDE Page Table
XD	Ignored	Rsvd.	Address of the 64KB MB page	Rsvd.		Ign	EA	Ign	G	PAT	D	A	PCD	PWT	U/S	R/W	P	PTE (64KB Page)
XD	Ignored	Rsvd.	Address of 4KB Page			Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P	PTE (4KB Page)

Each table entry is further broken down along with the required functions. GFX has a 4-level page table which is pointed out by context descriptor starting with the 4th level of PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required for support.

Page walk in advanced mode with 48b VA requires 4 levels. The walk will start with a PML4 table pointer extracted from PASID entry and uses the 48b VA as index to consecutive levels of page tables.

The following diagram shows the page walk that is needed for a 4KB page:



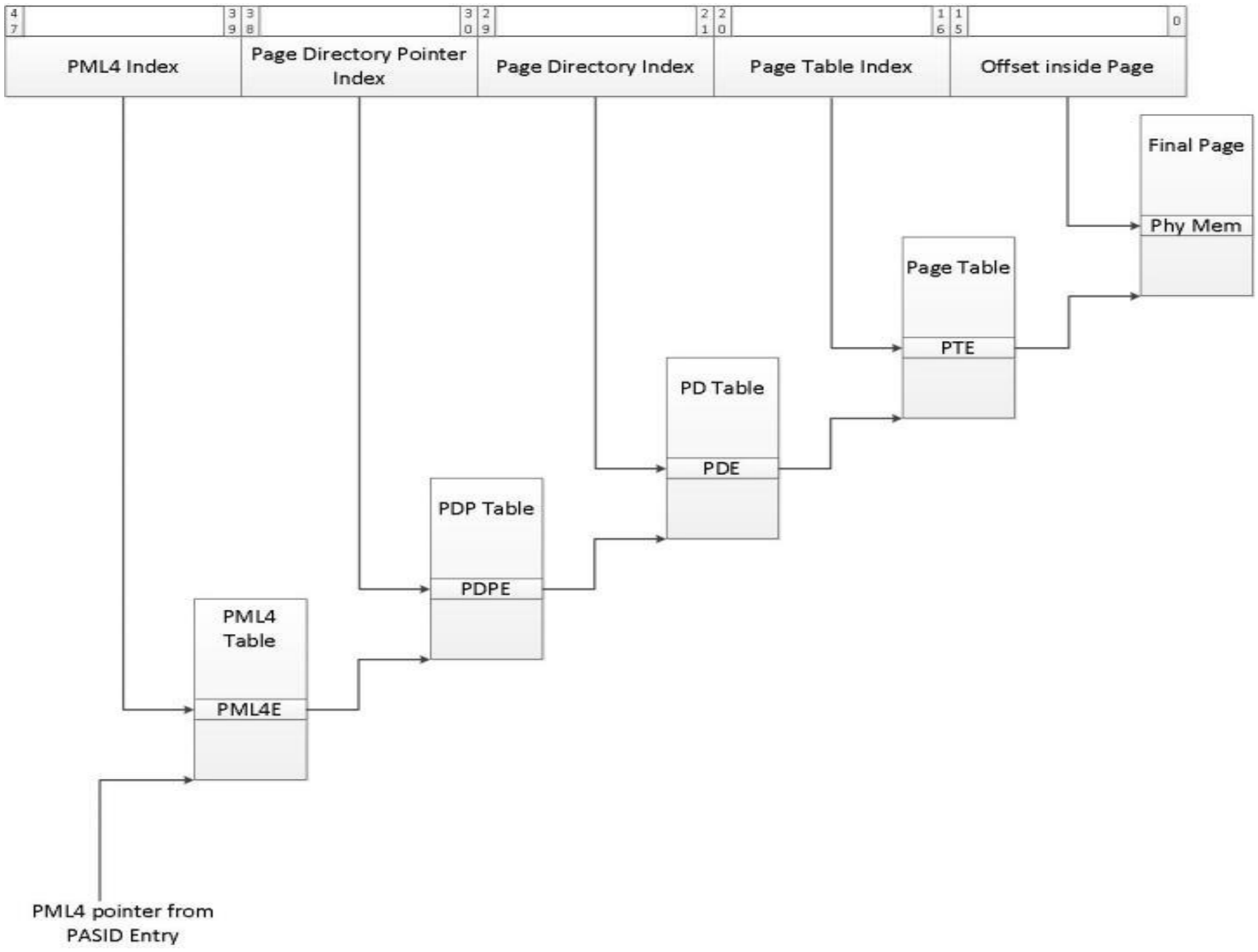
A 64 bit (48b canonical) address requires 4 levels of page table format where the context carries a pointer to highest level page table (PML4 pointer) via PASID. The rest of the walk is normal page walk thru various levels.

To repurpose the caches the following mechanism is used:

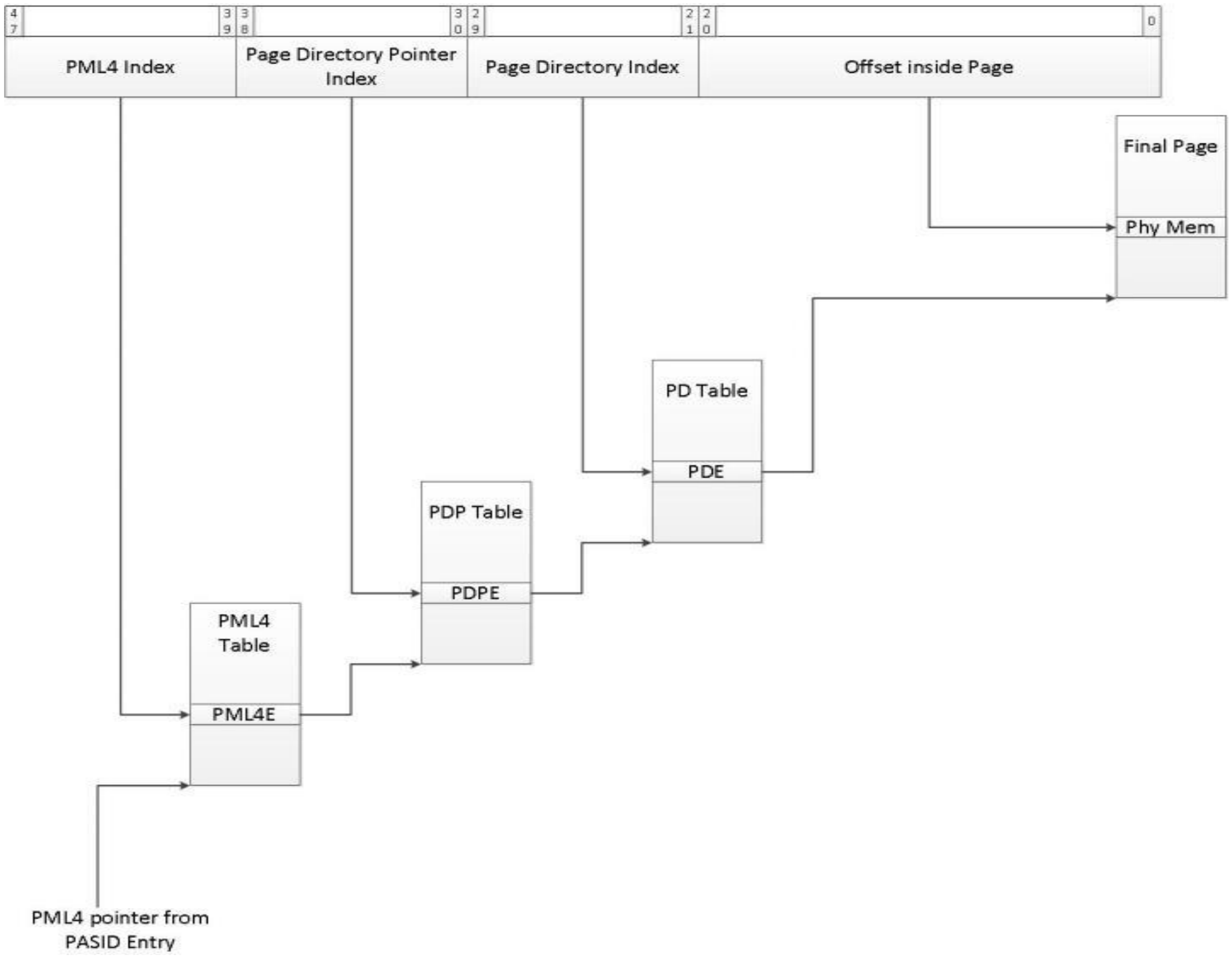
- 3D: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache.
- Media: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache.
- VEBOX, Blitter: each with 4KB acting as PML4, PDP, PD cache.

The design sections the 512 entries within 4KB into separate areas for PML4, PDP, and PD.

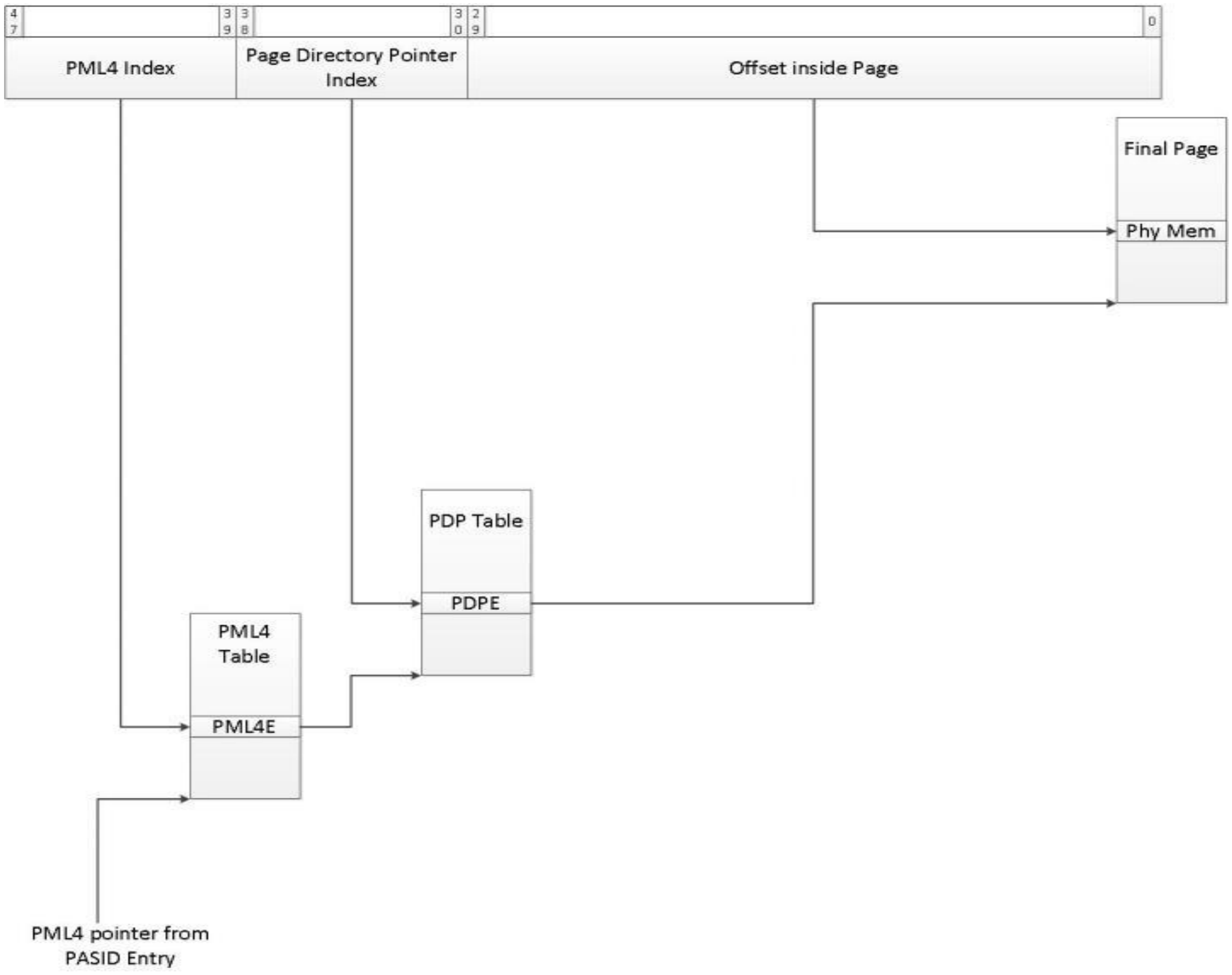
The 64KB Page size has a slightly different usage for how PTEs are selected for the corresponding 64KB page. In a page table every 16th entry (PTE#0, PTE#16, PTE#32, ... PTE#496) should be used to index. This is calculated using address[20:16] & “0000”. Note that hardware should not make any assumptions for any other PTEs.



With the 2MB Page walk, the last level of the page walk is skipped where the PD entry points to the final page.



For the support for 1GB page size, the following mechanism is needed.



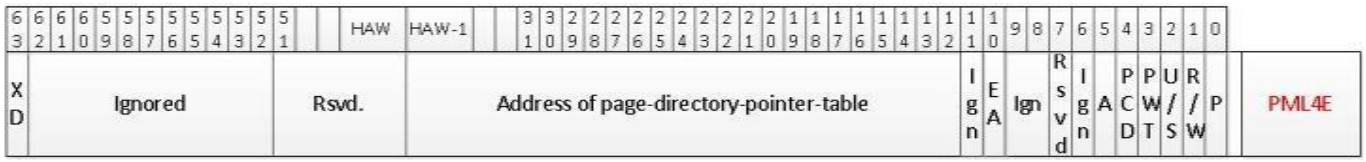
Pointer to PML4 Table

Page table pointer is the starting address where the PML4 table starts. The contents of pointer will be provided by PASID table entry in case of advanced context, else it will be provided by software as part of the legacy context with 48b addressing.

Details of PASID entry is given in later sections.

PML4E: Pointer to PDP Table

PML4 is used to locate the page directory pointer tables distributed in physical memory. For gen8/9, PML4 will be used for advanced GPGPU context scheduled via PASID table as well as legacy context with 48b VA.



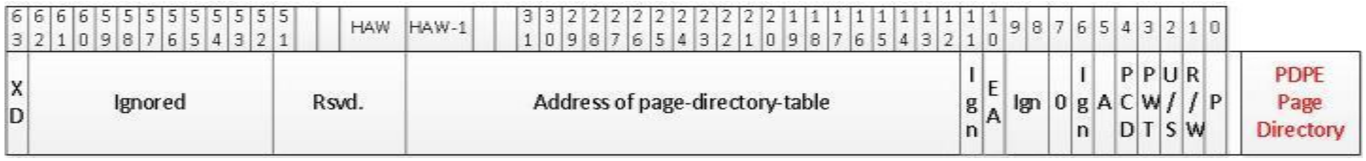
Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. *Not support in gen9*
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9:8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	*Reserved*	*Reserved (must return 0's)*
6	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. *GPU does not support any memory type but WB when accessing paging structures.*
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry. *GPU does not support any memory type but WB when accessing paging structures.*
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights.

Bits	Field	Description
		GPU does not support Supervisor mode contexts.
0	P: Present	PML4 Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE: Pointer to PD Table

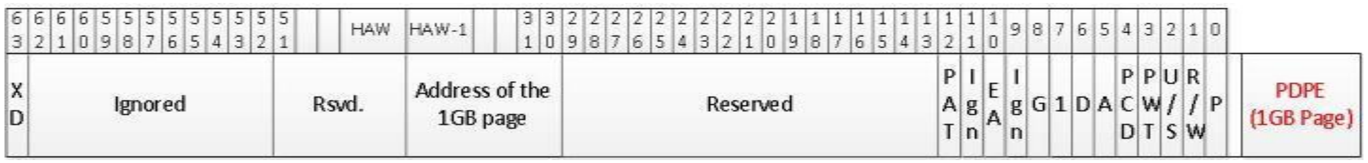
PDP entry is used to locate the base of the PD table:



Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. *Not support in gen9*
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9:8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	*Reserved*	*Reserved (must return 0's)*
6	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. *GPU does not support any memory type but WB when accessing paging structures.*
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.

Bits	Field	Description
		GPU does not support any memory type but WB when accessing paging structures.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. It must be “1” to point to a page directory pointer table

PDP entry for 1 GB Page



Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. *Not support in gen9*
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):30	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
29:13	Reserved	*Reserved (must return 0's)*
12	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	Page Size	Must be 1 to indicate 1GB page.

Bits	Field	Description
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be “1” to point to a 1GB Page.

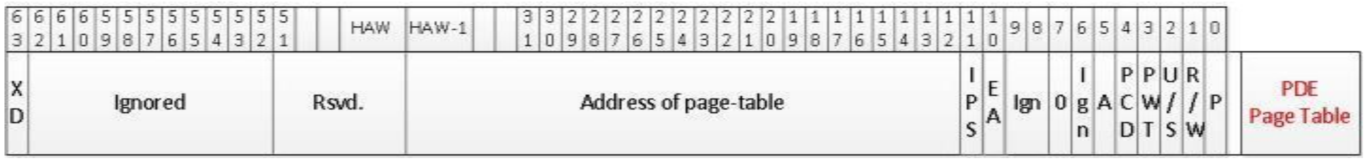
* HAW = 39 for client, and 46 for server.

PD: Pointer to Page Table

Page Directory entry has few different usage models:

1. It can identify the base of the page table.
2. It can define 2MB page table entries.

Pointer to page table is given below:

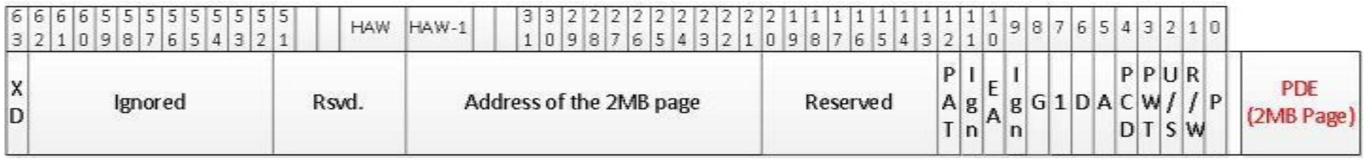


63	62 … 52	51 … HAW	HAW-1 … 12	11	10	9 8	7	6	5	4	3	2	1	0	
XD	Ignored	Rsvd.	Address of page-table	IPS	EA	Ign	0	Ign	A	PCD	PWT	U/S	R/W	P	PDE Page Table

Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. *Not support in gen9*
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	IPS	An MMIO level control has been introduced to manage 64KB page enabling.
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9:8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	*Reserved*	*Reserved (must return 0's)*
6	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. *GPU does not support any memory type but WB when accessing paging structures.*
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry. *GPU does not support any memory type but WB when accessing paging structures.*
2	U/S:	User vs supervisor access rights. If 0, requests with user-level privilege are not

Bits	Field	Description
	User/Supervisor	allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PD Entry is present. It must be "1" to point to a page directory pointer table

PDE for 2MB Page is given below:

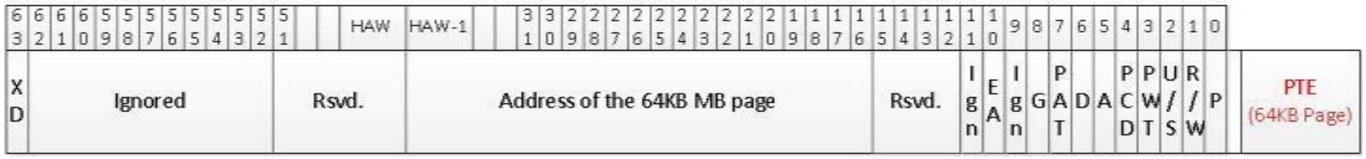


Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. *Not support in gen9*
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):21	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
20:13	Reserved	*Reserved (must return 0's)*
12	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	Page Size	Must be 1 to indicate 2MB page.
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.

Bits	Field	Description
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 2MB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 64KB Page



63 62 61 60 59 58 57 56 55 54 53 52 51 HAW HAW-1 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
XD
Ignored
Rsvd.
Address of the 64KB MB page
Rsvd.
Ign
EA
Ign
G
PAT
D
A
PCD
PWT
U/S
R/W
P
PTE
(64KB Page)

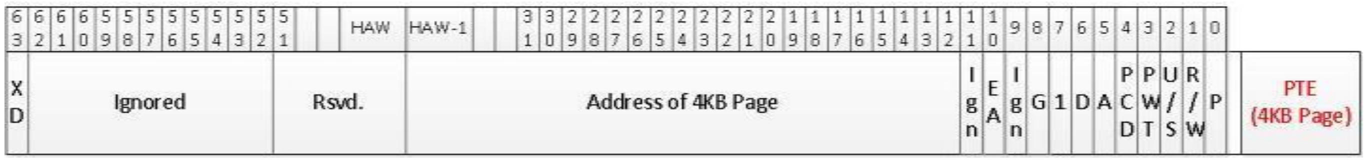
Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. *Not support in gen9*
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:12	Reserved	*Reserved (must return 0's)*
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this

Bits	Field	Description
		bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 64KB Page.

64KB pages need to be enabled via MMIO along with the PDE IPS bit per directory entry.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 4KB Page



63	62 61 60 59 58 57 56 55 54 53 52	51 HAW	HAW-1 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12	11	10	9	8	7	6	5	4	3	2	1	0	
XD	Ignored	Rsvd.	Address of 4KB Page	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P	PTE (4KB Page)

Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. *Not support in gen9*
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights.

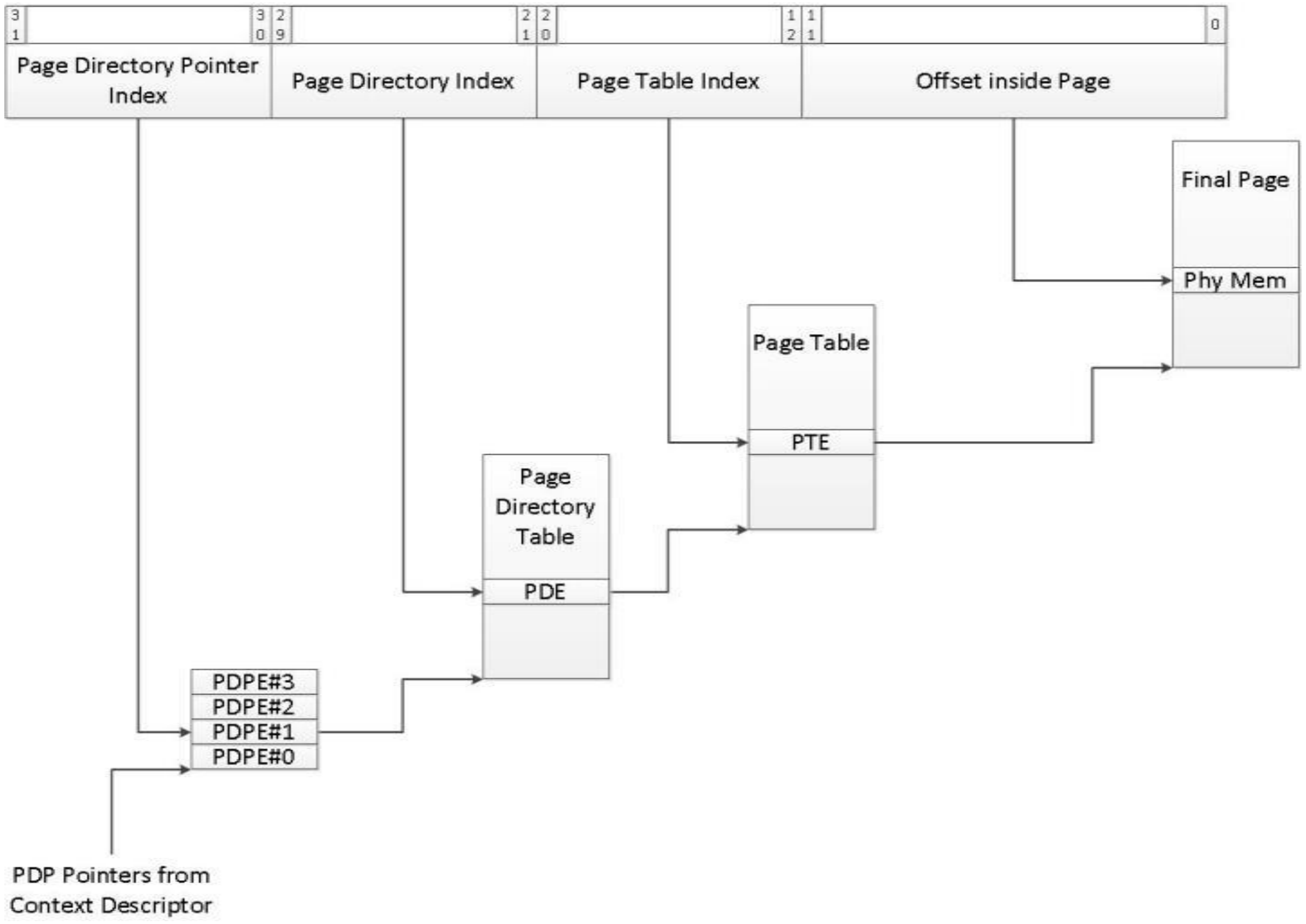
Bits	Field	Description
		GPU does not support Supervisor mode contexts.
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 4KB Page.

* HAW = 39 for client, and 46 for server.

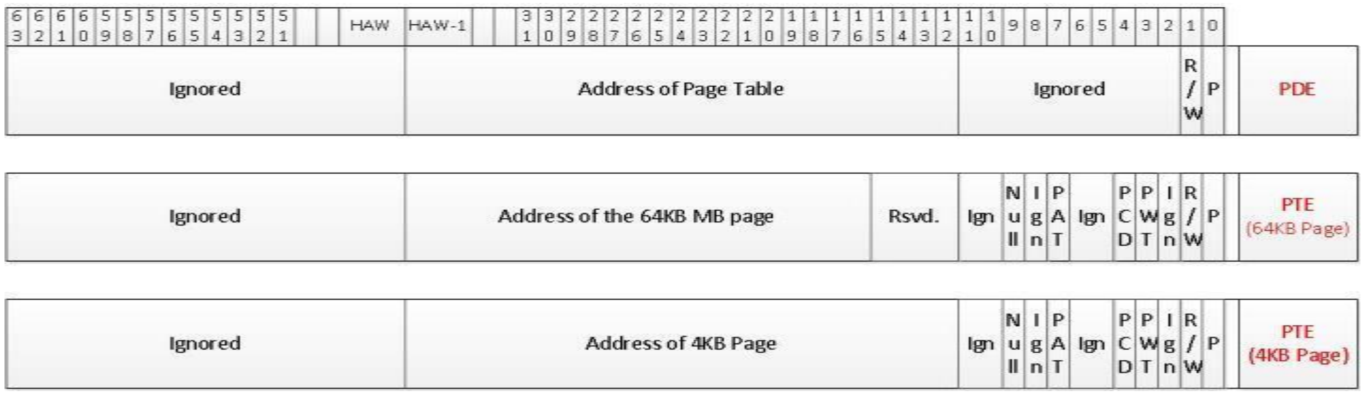
PPGTT for 32b Virtual Address

For page walk in legacy mode with 32b VA, we need two levels. The walk starts with a PDP pointer provided by the context descriptor, and uses the 32b VA as an index to consecutive levels of page tables. Hardware implements 16KB intermediate caches to limit the page walk needed to a single level, to have the same sensitivity to latency as previous generations.

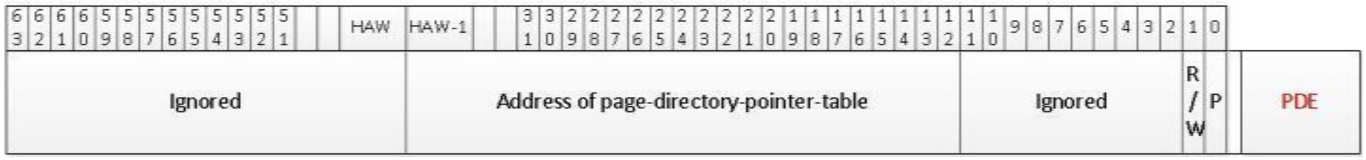
The following diagram shows the page walk needed for a 4KB page.



Page Table Entry formats for 32b VA use the following formats:

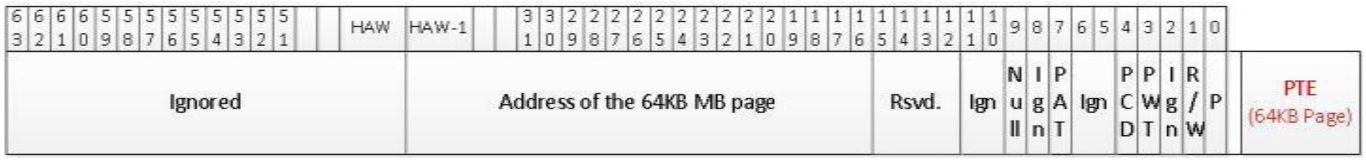


PDE for the page table



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests targeted to the memory range pointed by this PDE. *In Legacy mode with 32b VA, R/W bits from PDE are not used.*
0	P: Present	PD Entry is present. It must be "1" to point to a page directory pointer table

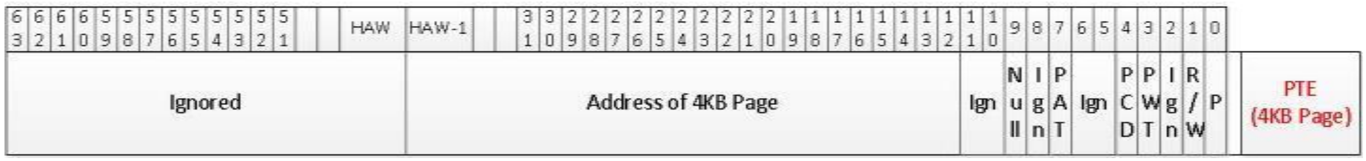
PTE for 64KB page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the

Bits	Field	Description
		memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 64KB Page.

PTE for 4KB Page



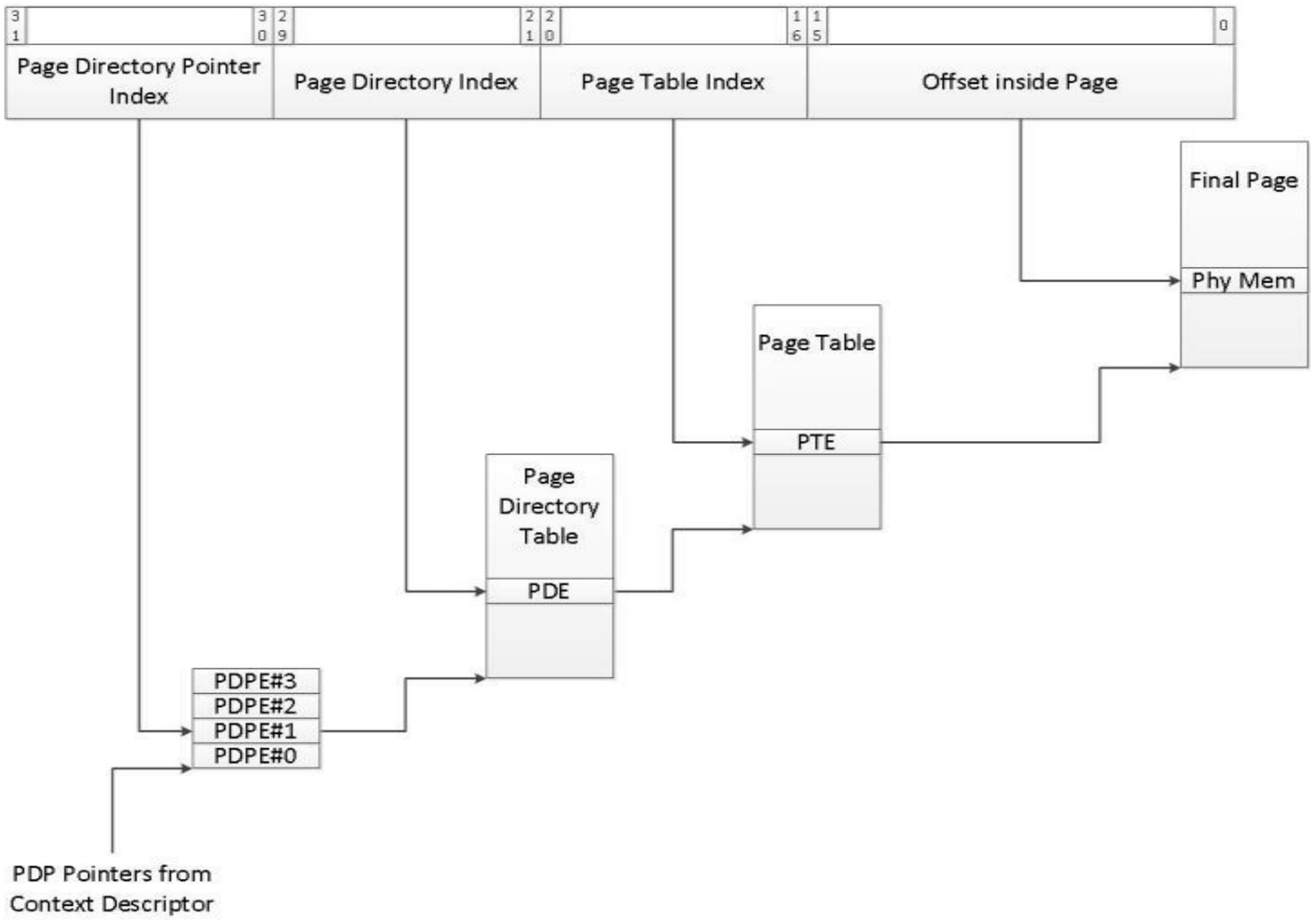
Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.

Bits	Field	Description
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 64KB Page.

* HAW = 39 for client, and 46 for server.

Walk with 64KB Page

64KB Page size has a slightly different usage for how PTEs are selected for the corresponding 64KB page. In page table every 16th entry (PTE#0, PTE#16, PTE#32....PTE#496) should be used to index. This is calculated using address[21:16] & "0000". Note that hardware should not make any assumptions for any other PTEs.



Walk with 2MB Page

PPGTT32 does not support 2MB pages.

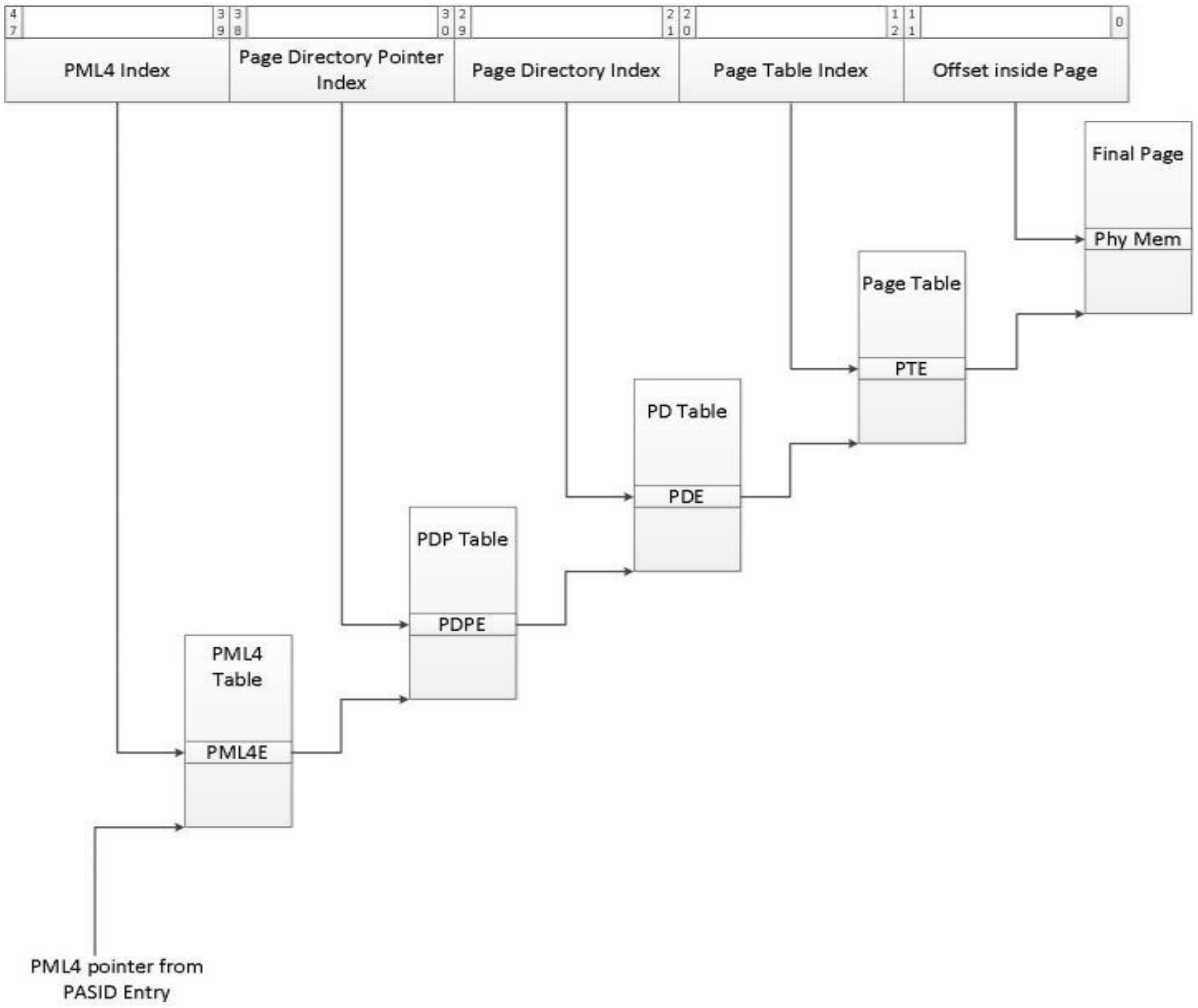
Walk with 1GB Page

PPGTT32 does not support 1GB pages.

PPGTT for Standard Context (64b VA)

For page walk in advanced mode with 48b VA, we need four levels. The walk starts with a PML4 table pointer given by GFX software and uses the 48b VA as index to consecutive levels of page tables.

The following diagram shows the page walk that is needed for a 4KB page:



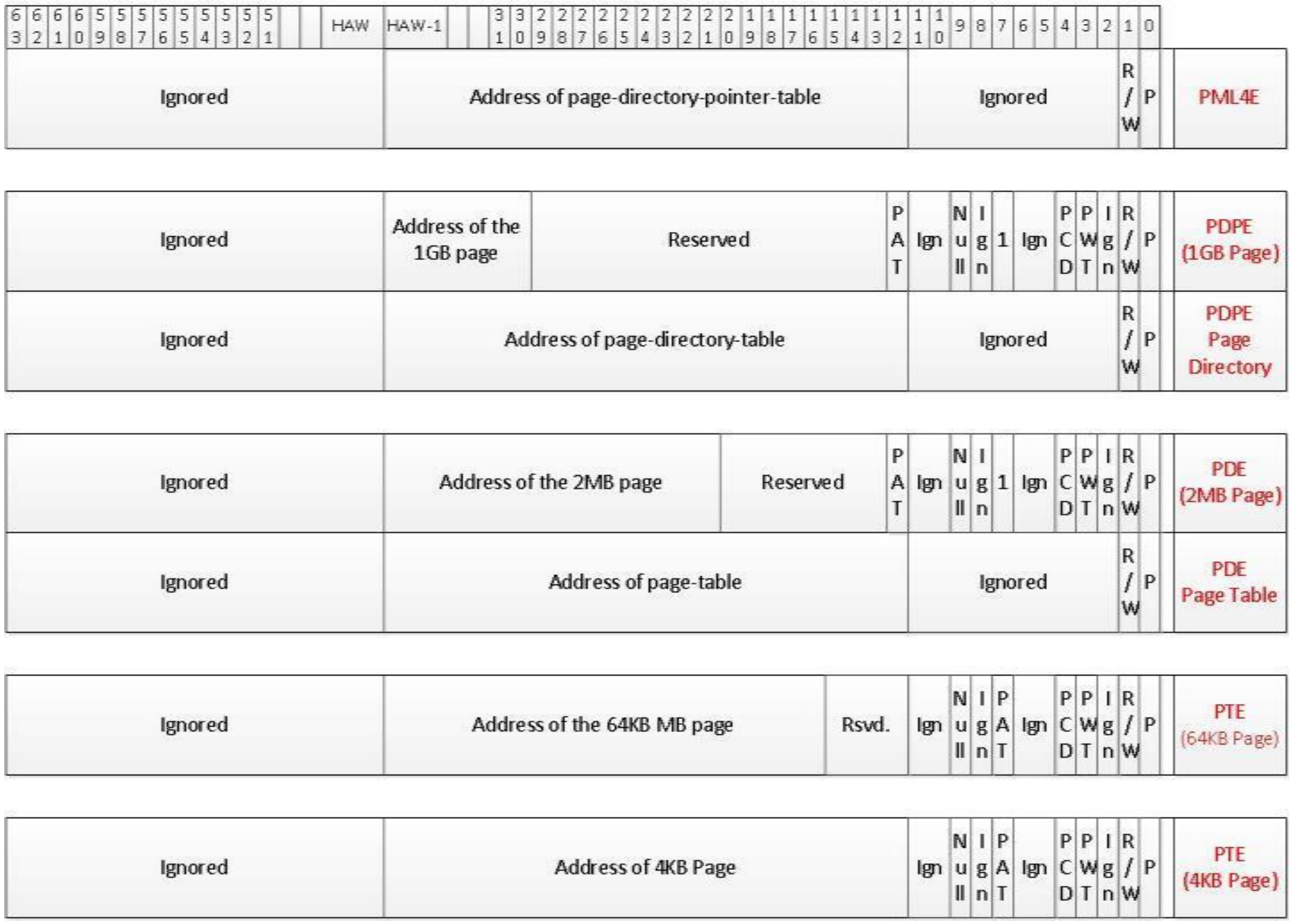
A 64-bit (48b canonical) address requires 4-levels of page table format where the context carries a pointer to the highest level page table (PML4 pointer) via PASID. The rest of the walk is a normal page walk thru the various levels.

To repurpose the caches the following mechanism is used:

- 3D: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache.
- Media: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache.
- VEBOX, Blitter: each with 4KB acting as PML4, PDP, PD cache.

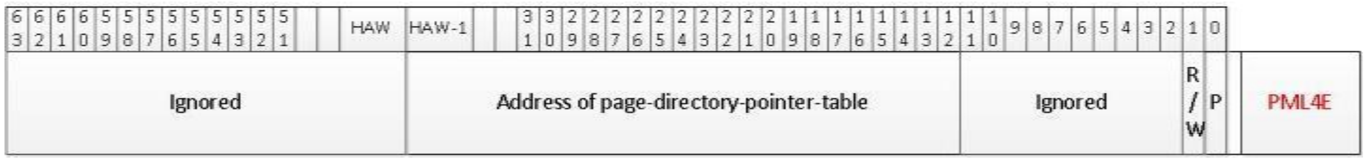
The design sections the 512 entries within 4KB into separate areas for PML4, PDP, and PD.

Page Table Entry (PTE) formats follow a similar layout to IA32e as given below.



Each table entry is further broken down along with the required functions. GFX has a 4-level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required for support.

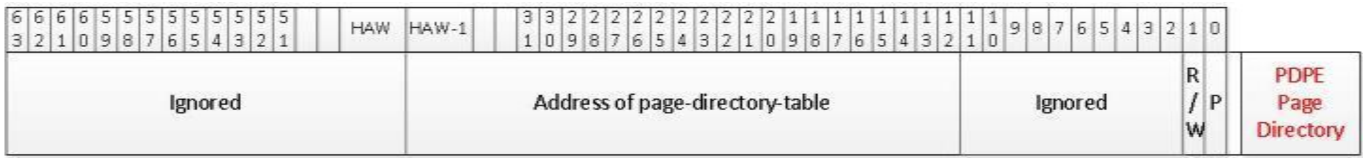
In 48b legacy mode, the pointer to the PML4 table is provided via the context descriptor provided by GFX software. The PML4 entry format is given below and points to the base of the PDP table.



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.

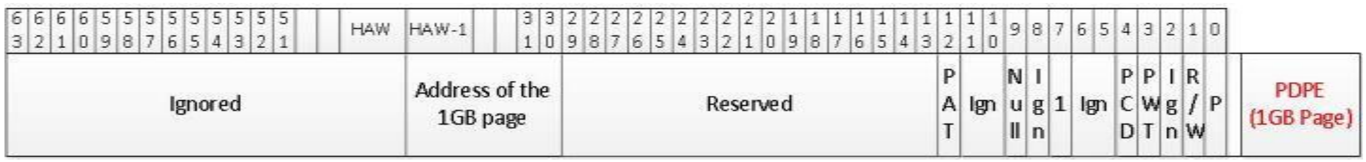
Bits	Field	Description
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.* *In 64b Legacy, R/W in PML4 entry can not be used for RO pages.*
0	P: Present	PML4 Entry is present. It must be "1" to point to a page directory pointer table

PDP entry is used to locate the page directory. Similar to IA32e page tables, legacy 48b VA supports 1GB pages, the PDPE has a mechanism to identify a way to say whether this PDPE represents a pointer to page directory or to a contiguous 1GB physical memory. PDP entry format is given below and points to the base of PD table.



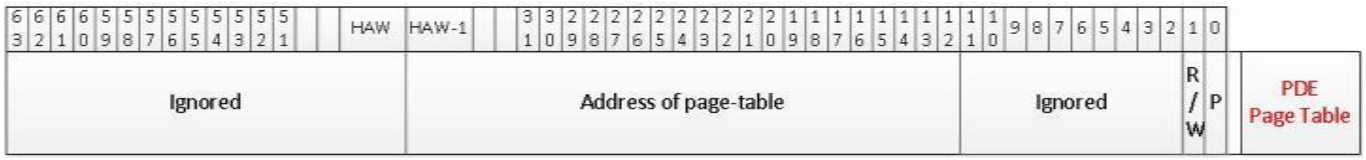
Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.* *In 64b Legacy, R/W in PDP entry can not be used for RO pages*
0	P: Present	PDP Entry is present. It must be "1" to point to a page directory pointer table

PDP entry for 1GB Page



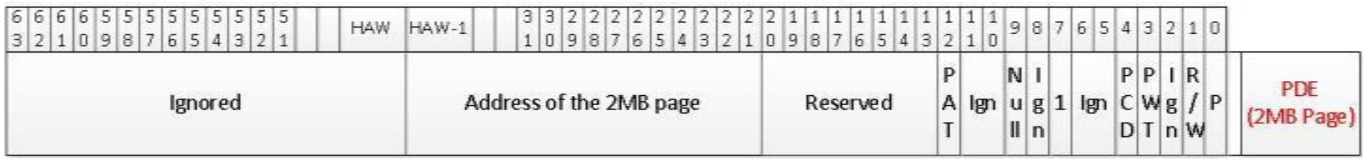
Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):30	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
29:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 1GB Page.

Page Directory entry point to the base of the page table and format is given below.



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page- table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.* *In 64b Legacy, R/W in PD entry can not be used for RO pages*
0	P: Present	PDP Entry is present. It must be "1" to point to a page directory pointer table

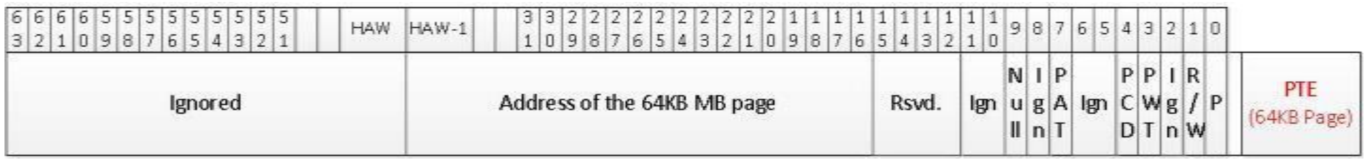
Page Directory entry for 2MB page:



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):21	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
20:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page	For devices operating in the processor coherency domain, this field indirectly

Bits	Field	Description
	level cache disable	determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 1GB Page.

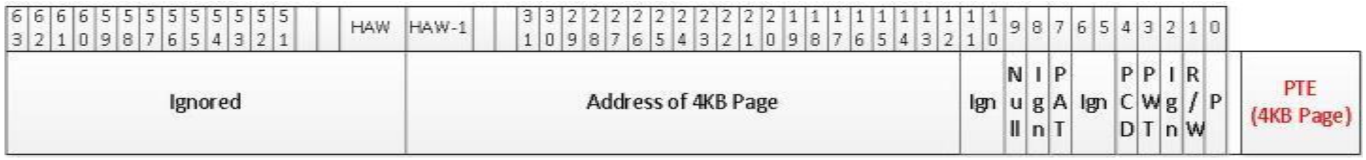
Page Table entry for 64KB page:



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*

Bits	Field	Description
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be “1” to point to a 64KB Page.

Page Table Entry for 4KB page:

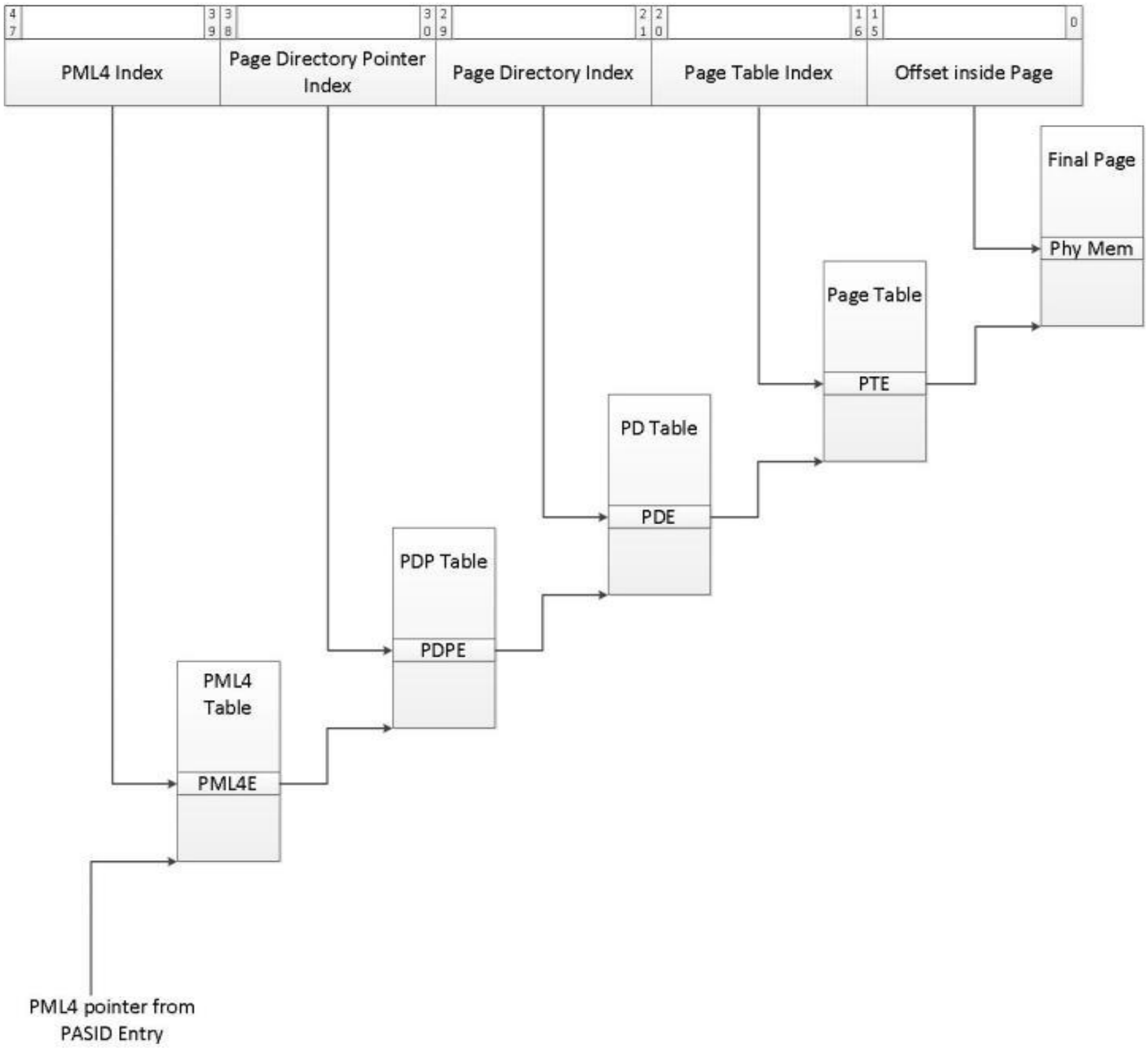


Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero’s for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be “1” to point to a 64KB Page.

* HAW = 39 for client, and 46 for server.

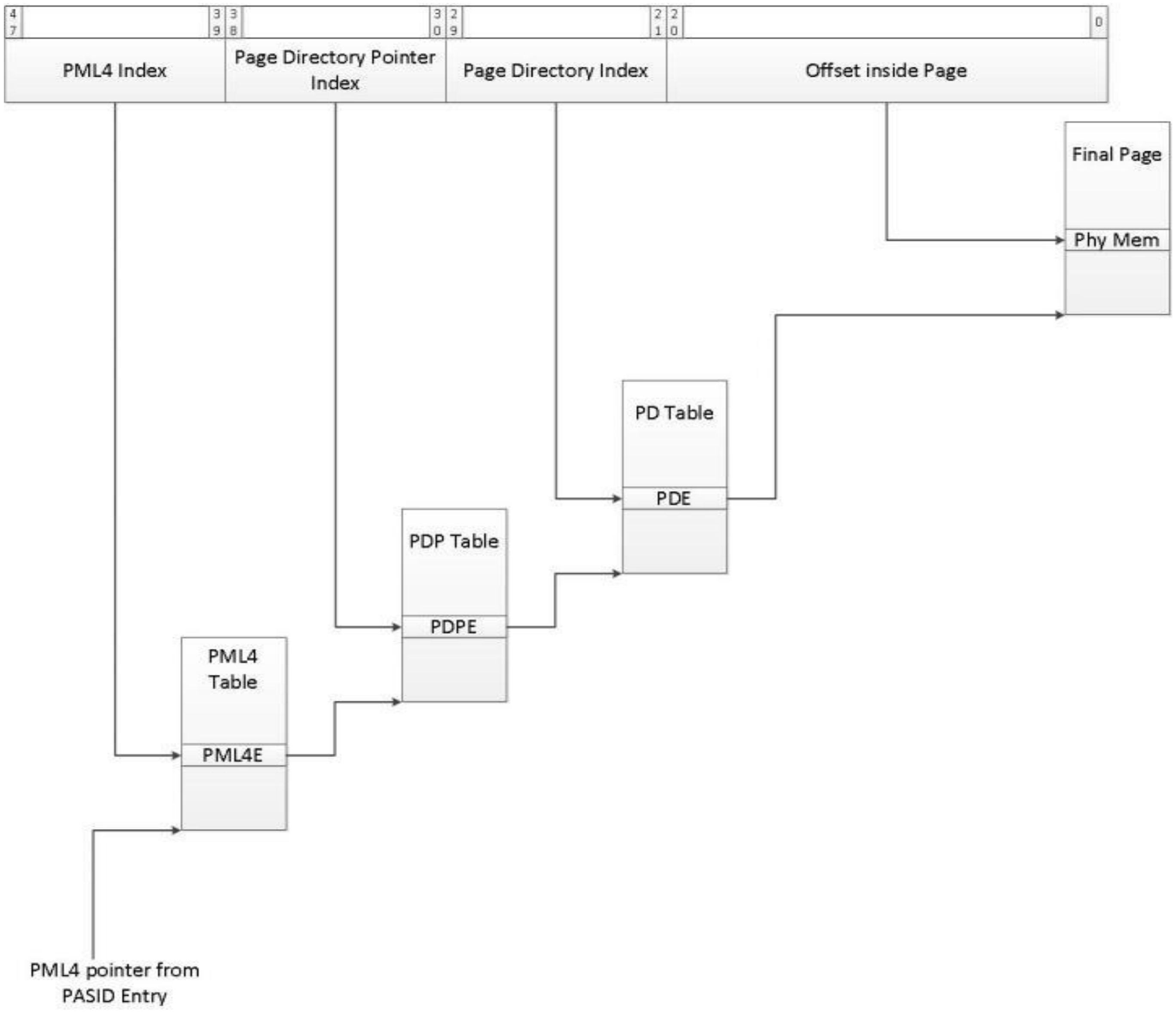
Walk with 64KB Page

64KB Page size has a slightly different usage for how PTEs are selected for the corresponding 64KB page. In page table every 16th entry (PTE#0, PTE#16, PTE#32....PTE#496) should be used to index. This is calculated using address [20:16]& "0000". Note that hardware should not make any assumptions for any other PTEs.



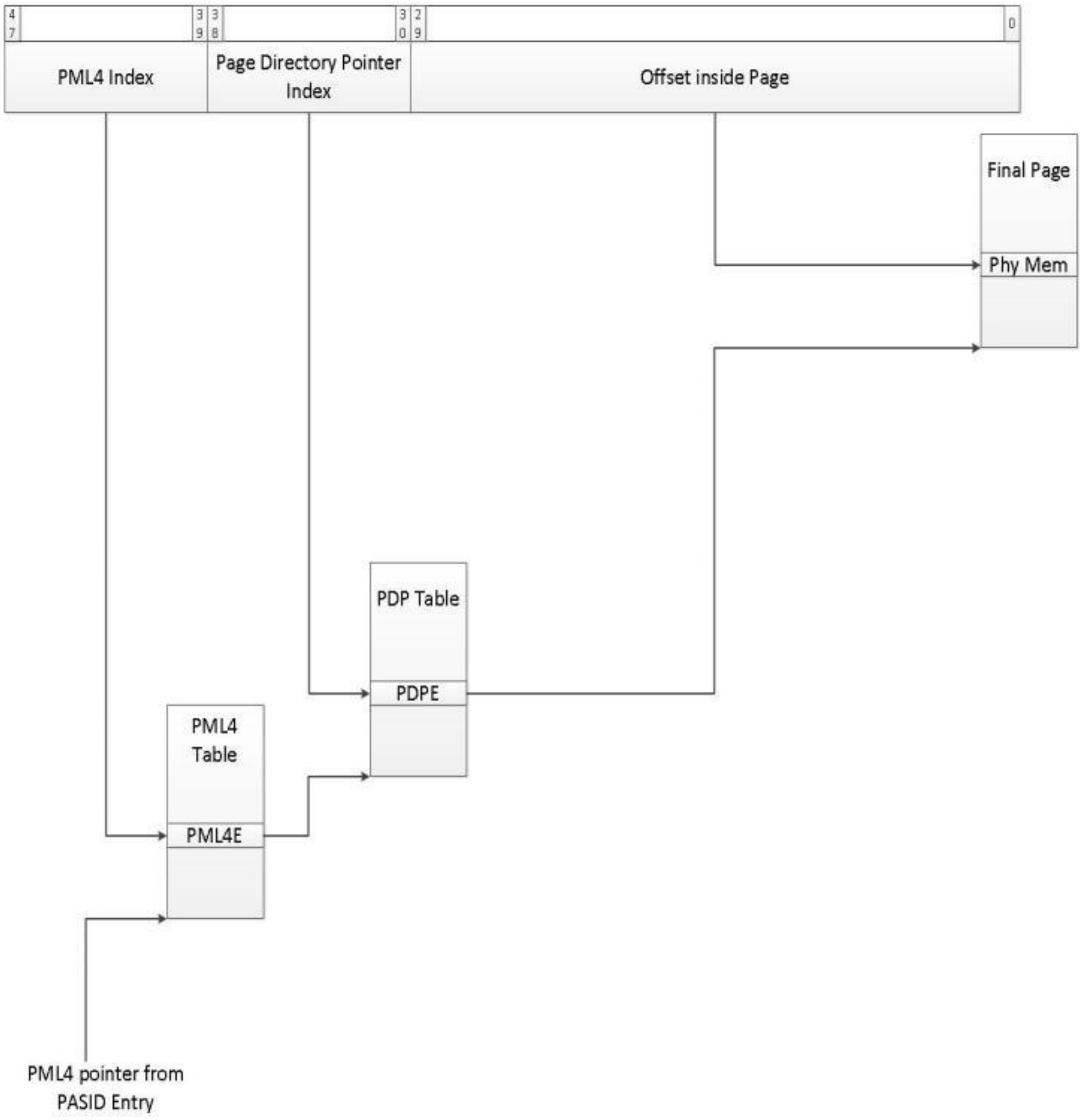
Walk with 2MB Page

With the 2MB Page walk, last level of the page walk is skipped where the PD entry points to the final page.



Walk with 1GB Page

For the support for 1GB page size, the following mechanism is needed.



Gen9 Global GTT

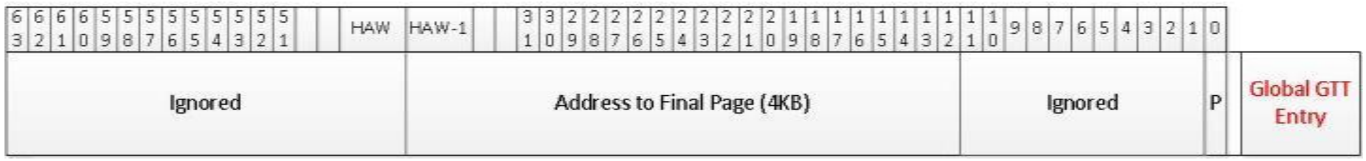
The Global GTT mechanism in gen8/9 looks very similar to pre-gen8 with the distinction of page table entry. Aperture and display will still use the global GTT even if GT core is mapped via per-process GTT.

The PTE format for Gen8/9 is updated to match per process GTT definitions and GSM is now expanded in size (2MB=>8MB) to cover for the entire 4GB (32b virtual addressing) space. Each entry corresponding to a 4KB page with 2^20 entries in GSM (each with 8B content)

For “*MI_update_GTT*”, the page address provided 31:12 need to be shifted down to 22:3 for the correct QW position within the GGTT.

Page Table Entry

The following page table entry will be used for Global GTT:



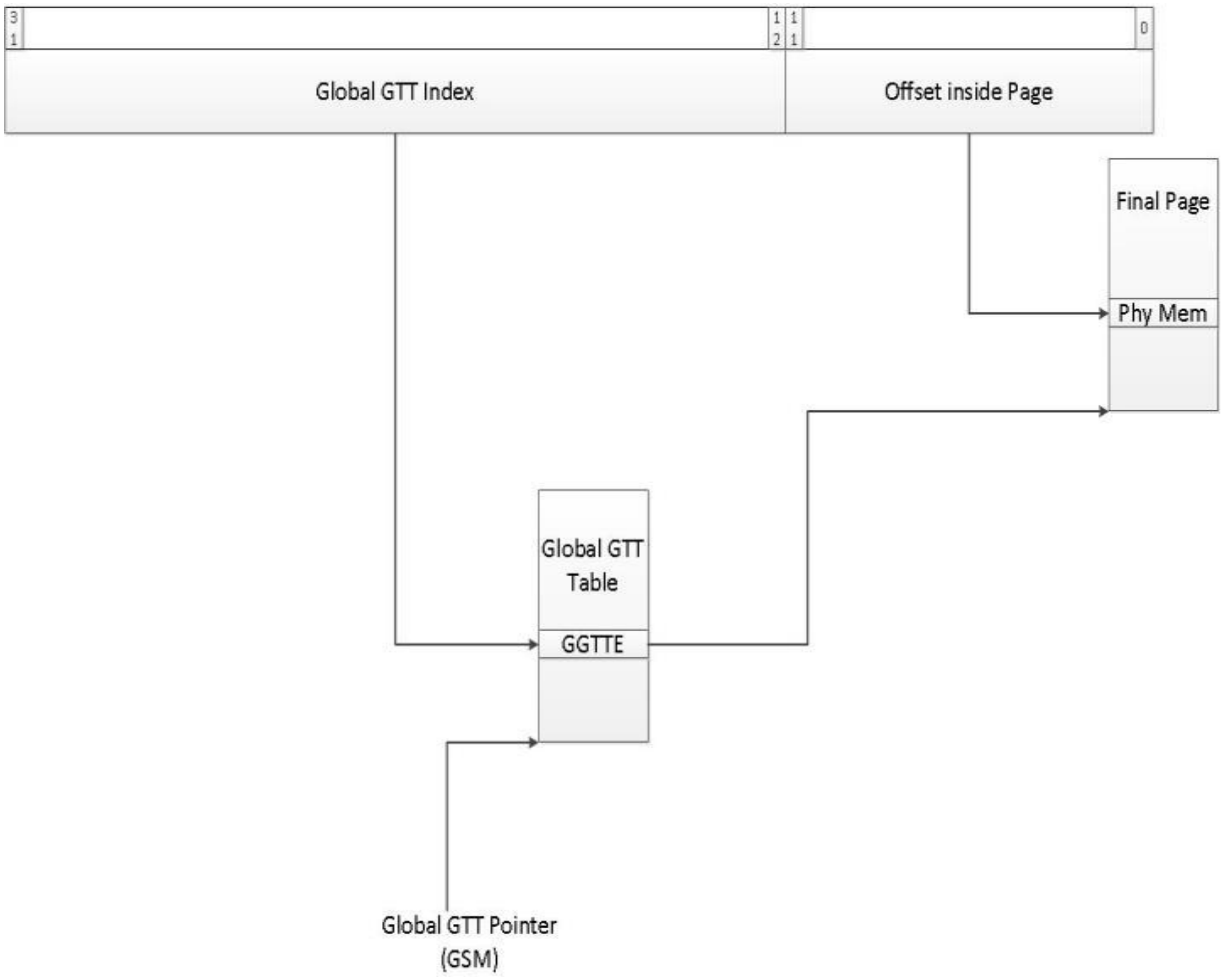
Bits	Field	Description
63:54	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	Address	Physical address of 4KB memory page referenced by this entry.
11:1	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
0	Present	When set to 1, indicates that this Page Table Entry is Valid, and the corresponding page is Present in physical memory

* HAW = 39 for client, and 46 for server.

The GPU accesses GGTT table entries as uncacheable.

Page Walk

The global GTT page walk is identical to what it was before gen8. The only difference would be that each entry is 8B (instead of 4B) hence the entry selection needs to be updated once the corresponding Page Table miss read is returned.



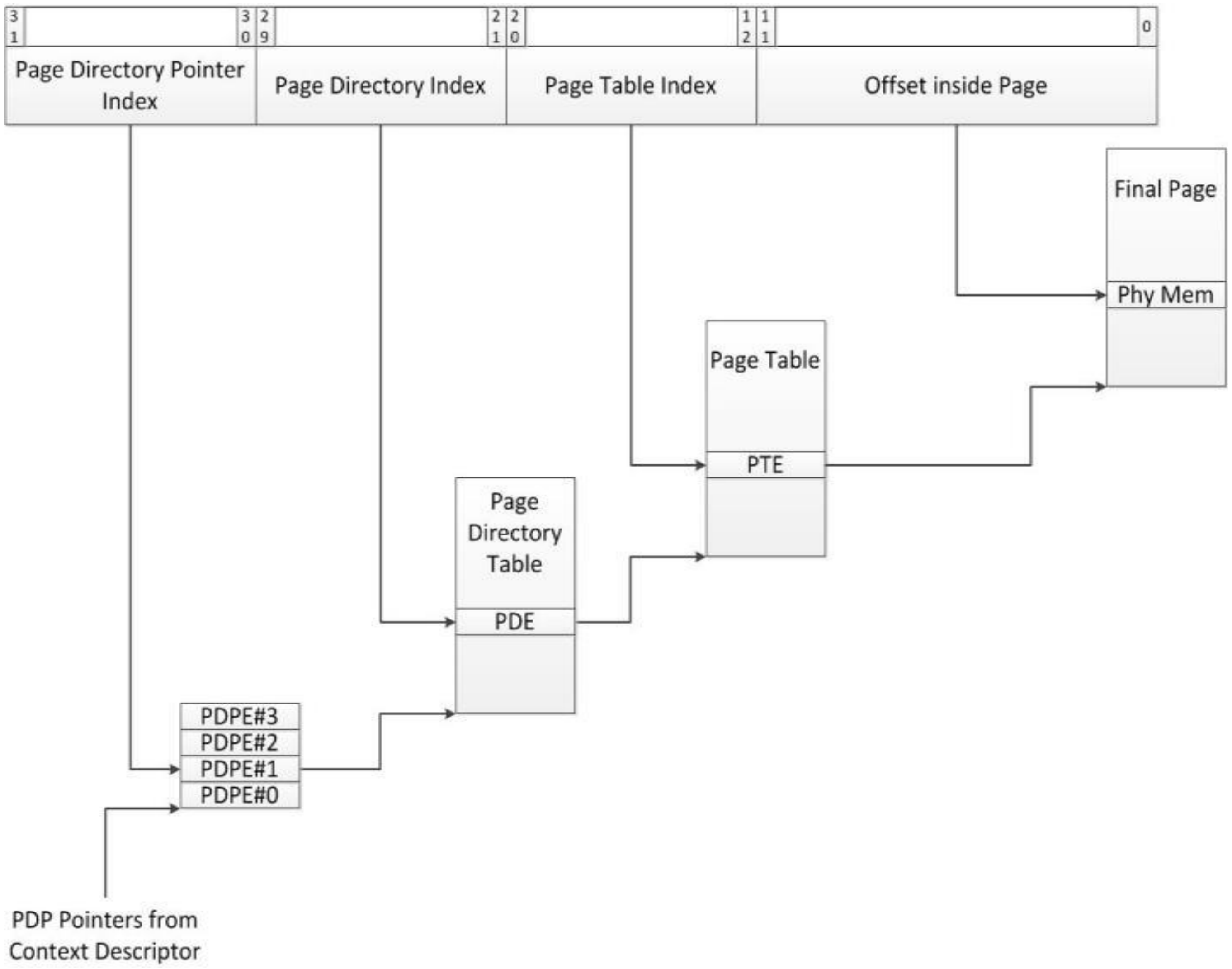
Legacy mode with 32b VA

Gen9 page walker is capable supporting 32b VA address with optimized page tables, this is to keep the walk to a single level.

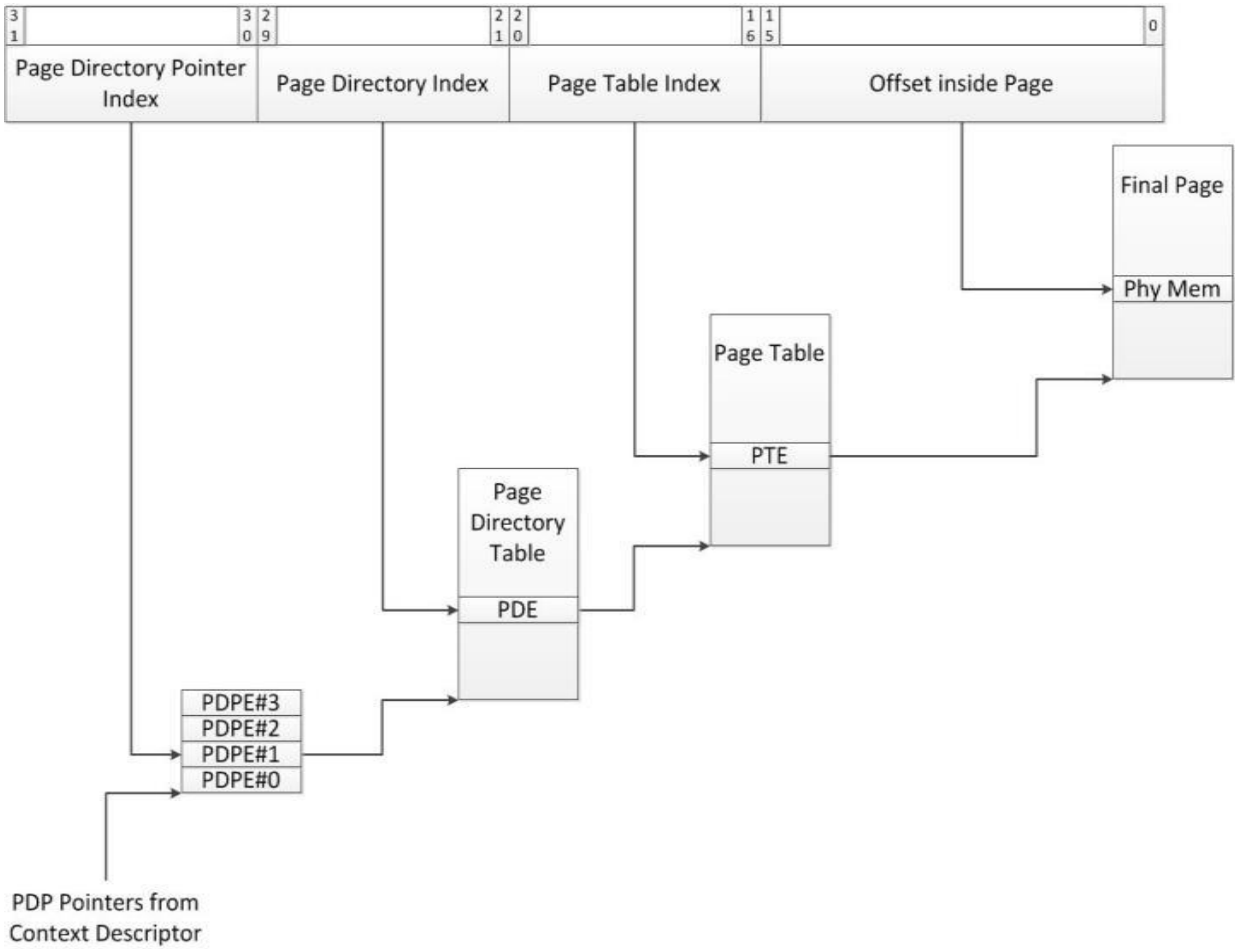
Page Walk in Legacy mode with 32b VA

For page walk in legacy mode with 32b VA, we need 2 levels. The walk will start with a PDP pointer provided by the context descriptor and uses the GraphicsAddress as an index to consecutive levels of page tables. Hardware implements 16KB intermediate caches to limit the page walk needed to a single level to have the same sensitivity to latency as previous generations.

The following diagram shows the page walk that is needed for a 4KB page.

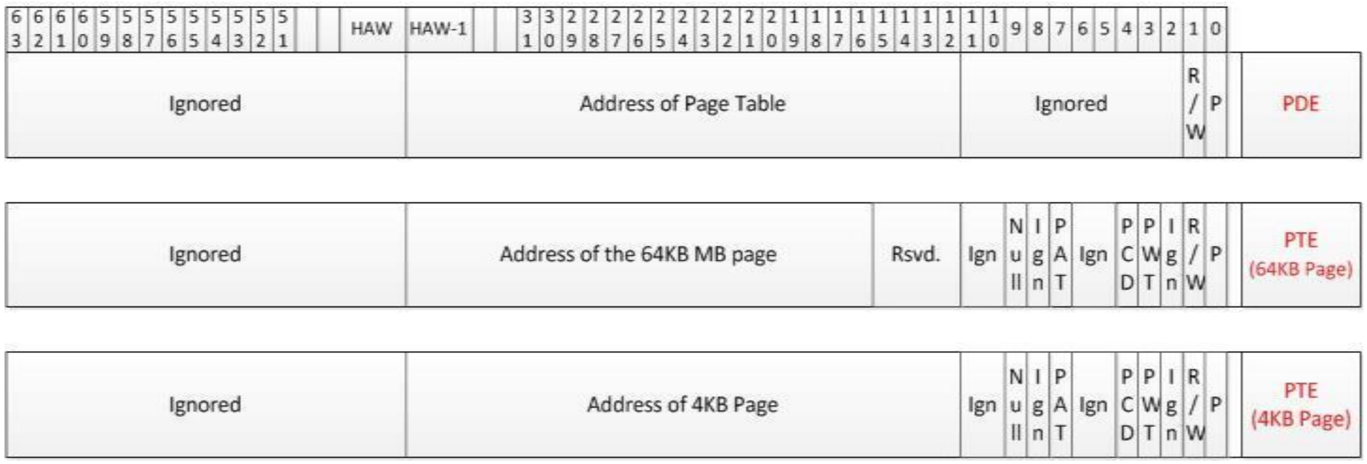

3
1
3
0
2
9
2
1
2
0
1
2
1
1
0
Page Directory Pointer Index
Page Directory Index
Page Table Index
Offset inside Page
Final Page
Phy Mem
Page Table
PTE
Page Directory Table
PDE
PDPE#3
PDPE#2
PDPE#1
PDPE#0
PDP Pointers from Context Descriptor

Walk with 64KB Page

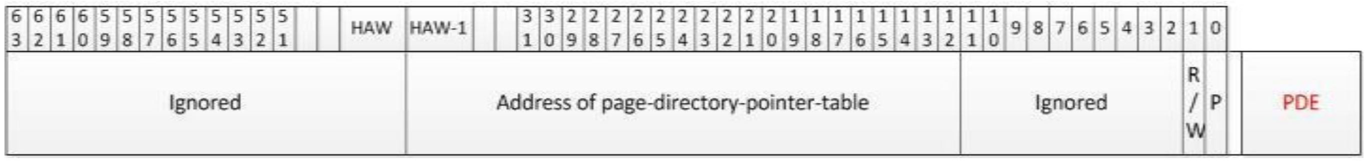


Page Table Entry (PTE) Formats

Page Table Entry formats for 32b VA use the following format:



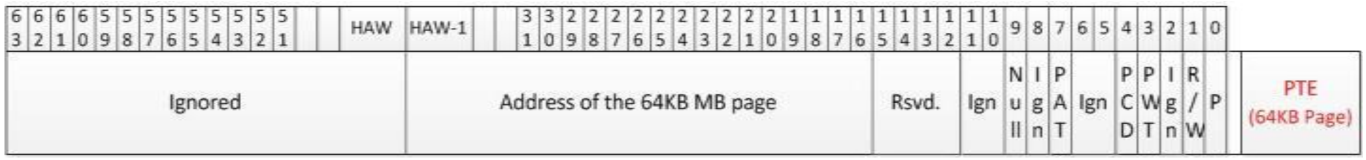
PDE for Page Table



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests targeted to the memory range pointed by this PDE. In Legacy mode with 32b VA, R/W bits from PDE are not used.
0	P: Present	PD Entry is present. It must be “1” to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 64KB Page

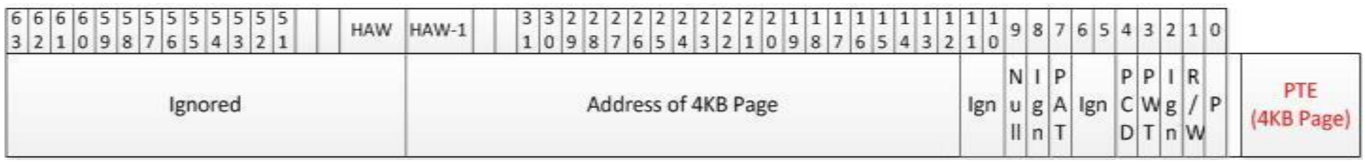


63 … 51	HAW	HAW-1 … 16	15 … 12	11 10	9	8	7	6 5	4	3	2	1	0	
Ignored		Address of the 64KB MB page	Rsvd.	Ign	Null	Ign	PAT	Ign	PCD	PWT	Ign	R/W	P	PTE (64KB Page)

Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 64KB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 4KB Page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of a 4KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	This bit must be "1" to point to a valid Page.

* HAW = 39 for client, and 46 for server.

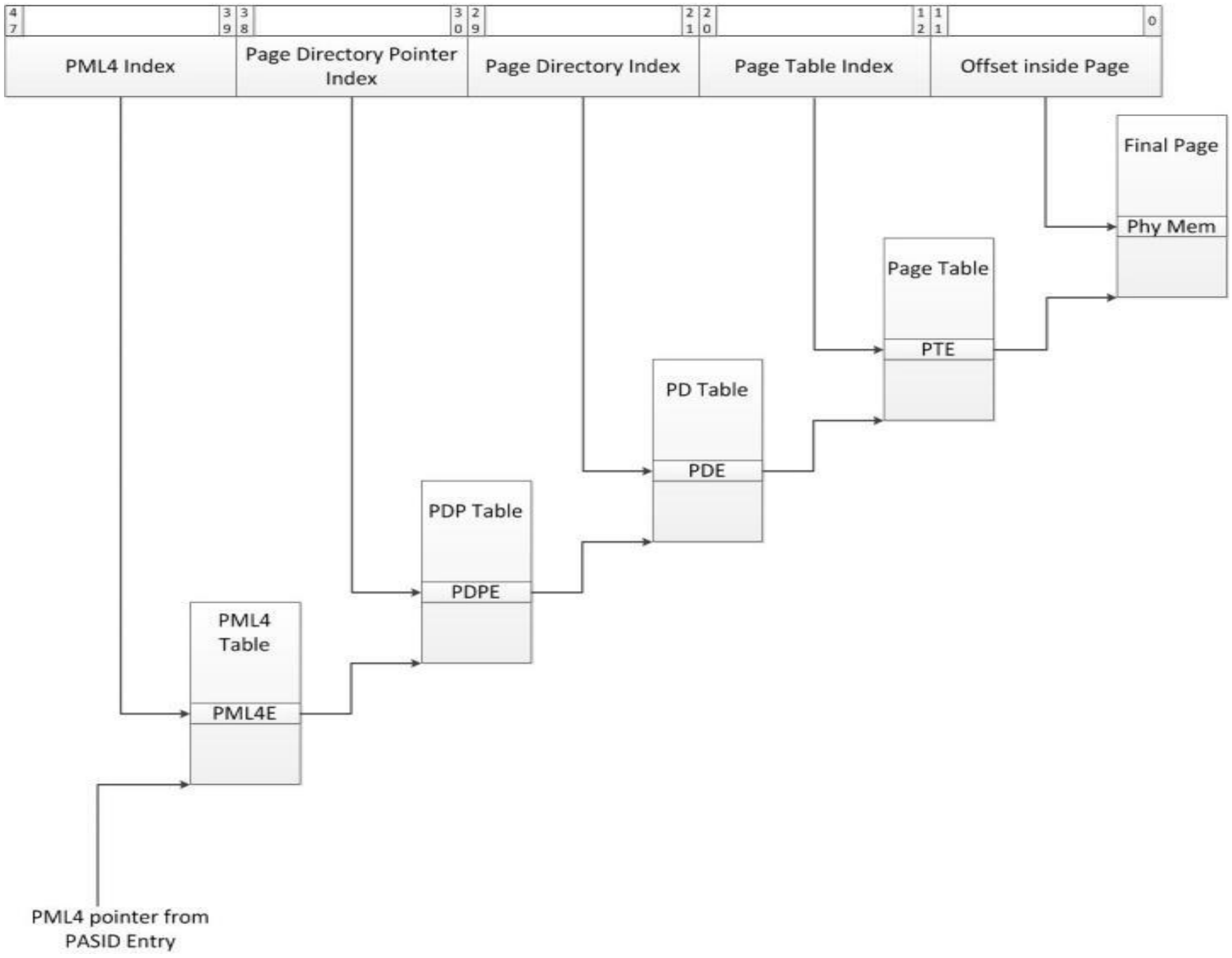
Legacy mode with 48b VA

Legacy mode with 48b VA enables larger virtual space while keeping the page walk compatible with IA32e.

Page Walk in Legacy 48b Mode

For page walk in advanced mode with 48b VA, we need 4 levels. The walk will start with a PML4 table pointer extracted from PASID entry and uses the 48b VA as index to consecutive levels of page tables.

The following diagram shows the page walk that is needed for a 4KB page.



64bit (48b canonical) address requires 4-levels of page table format where the context carries a pointer to highest level page table (PML4 pointer) via PASID. The rest of the walk is normal page walk thru various levels.

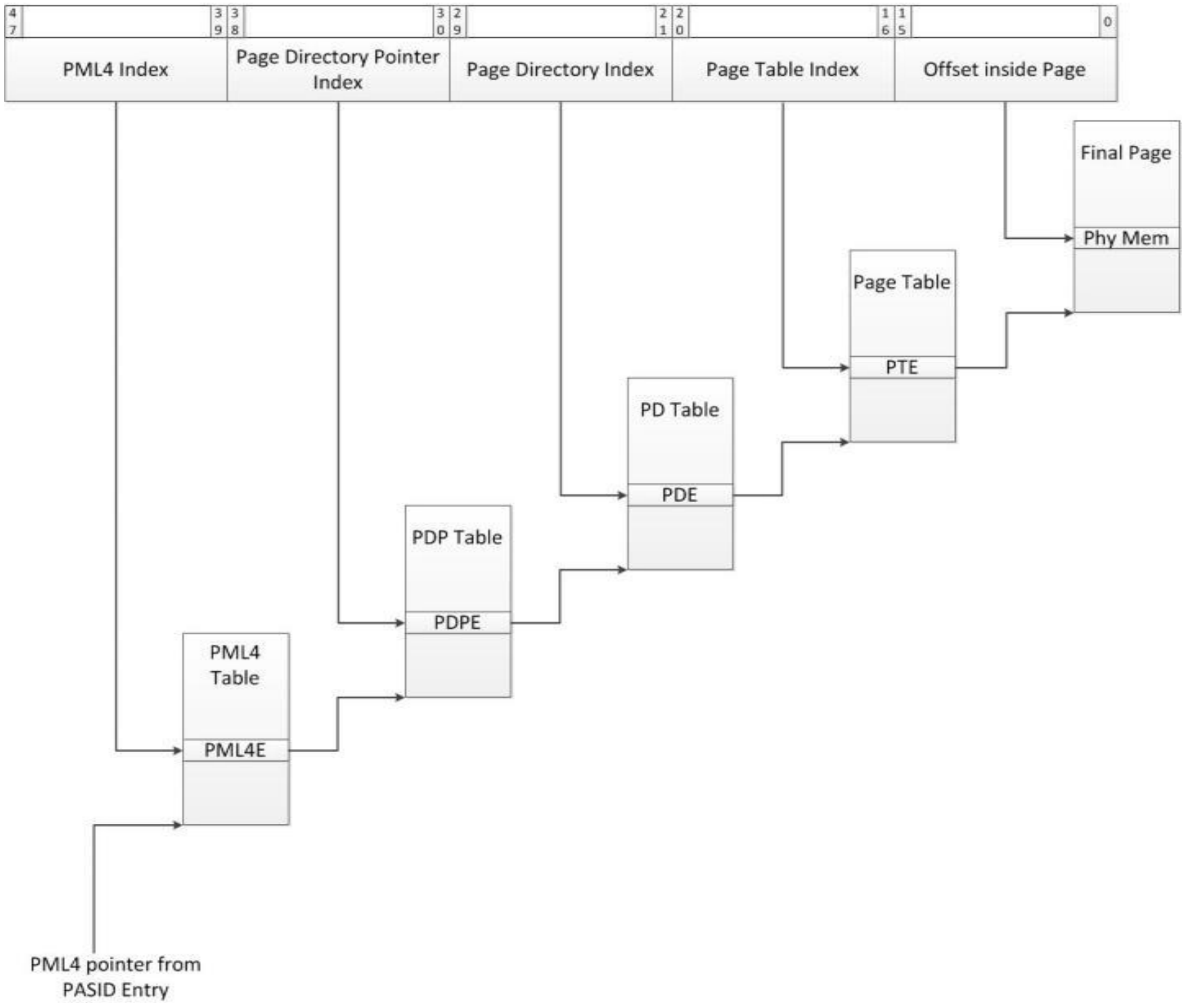
To repurpose the caches the following mechanism will be used:

- 3d: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache
- Media: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache
- VEBOX, Blitter: each with a 4KB acting as PML4, PDP, PD cache.

Note: design can section the 512 entries within 4KB to separate areas for PML4, PDP and PD.

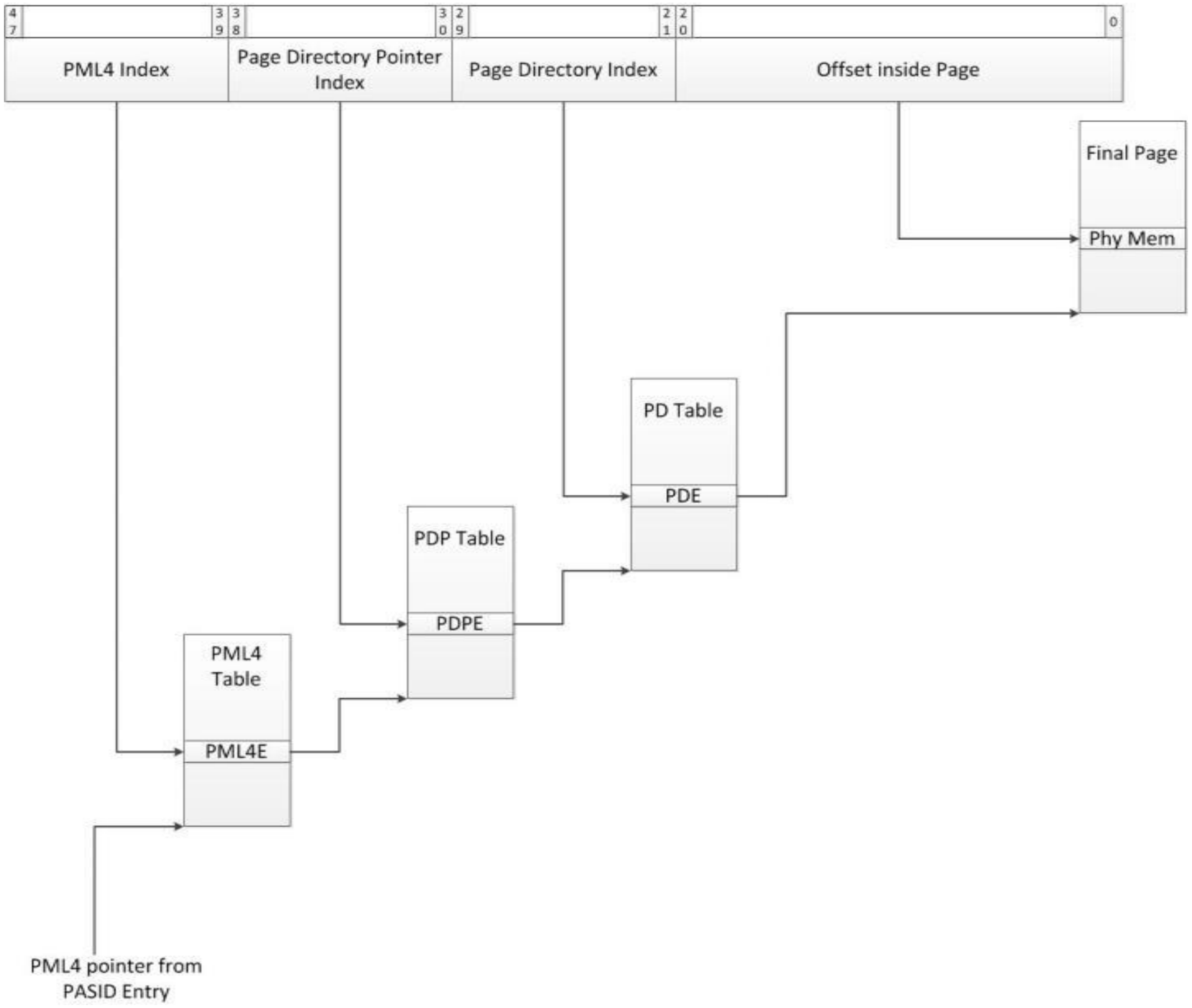
Walk with 64KB Page

64KB Page size has a slightly different usage for how PTEs are selected for the corresponding 64KB page. In page table every 16th entry (PTE#0, PTE#16, PTE#32....PTE#496) should be used to index. This is calculated using address [20:16]& "0000". Note that hardware should not make any assumptions for any other PTEs. 64K paging in the PTE is indicated by [11] of PDE. When PDE[11] = '1, every 16th PTE entry is read (by masking Adr[15:12] bits).



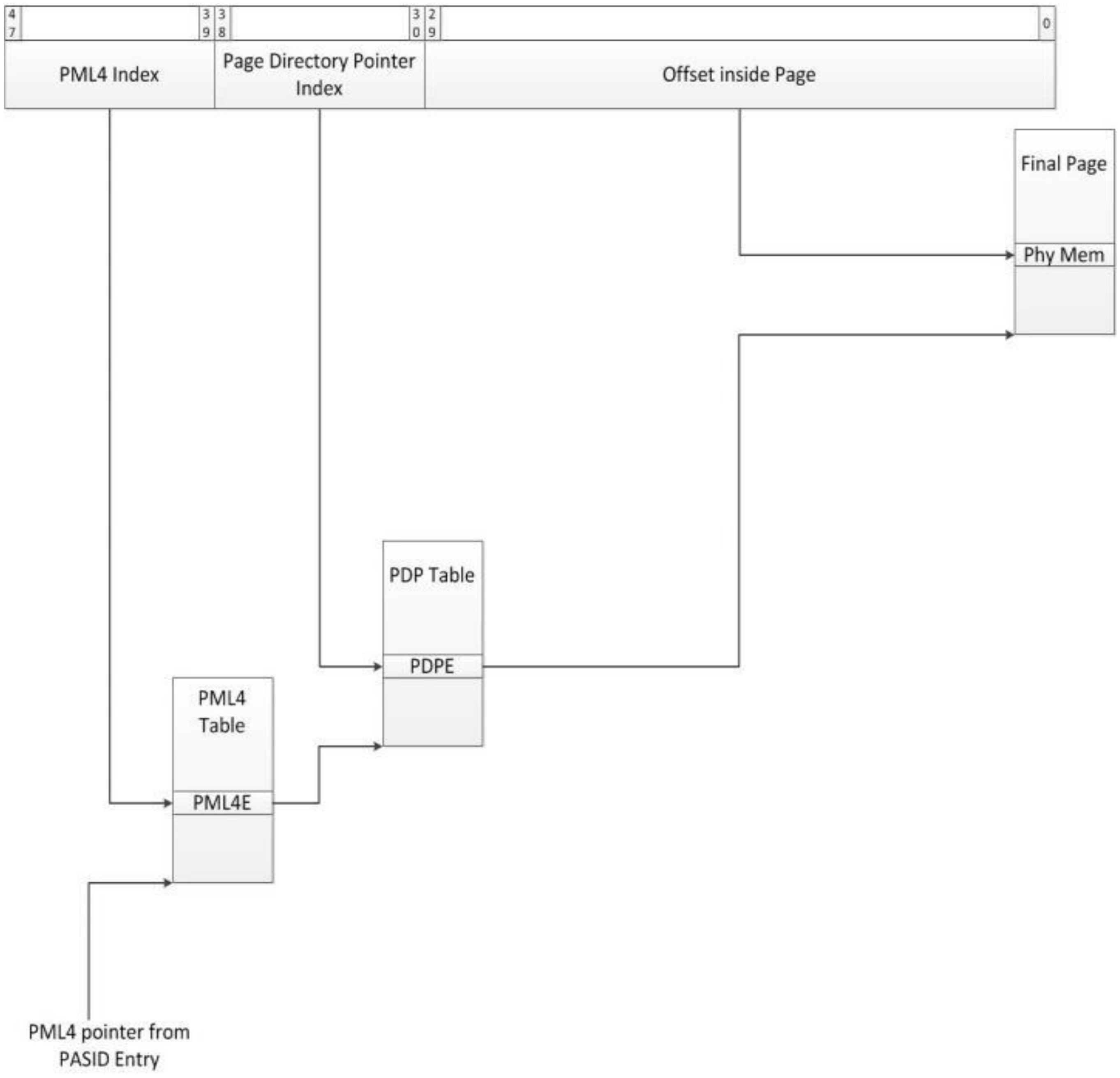
Walk with 2MB Page

With the 2MB Page walk, last level of the page walk is skipped where the PD entry points to the final page.



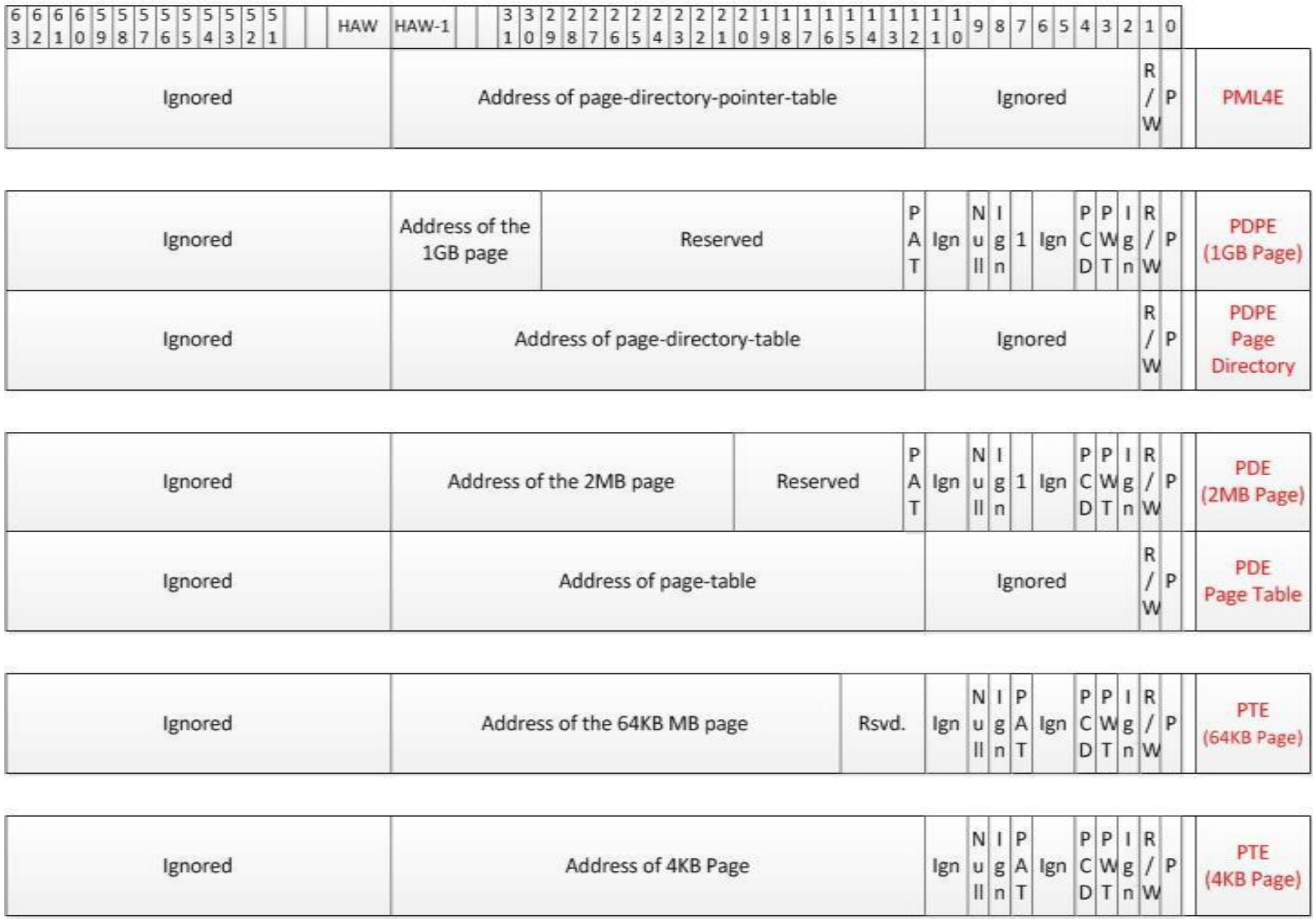
Walk with 1GB Page

For the support for 1GB page size, the following mechanism is needed.



Page Tables Entry PTE Formats

Page Table Entry (PTE) formats will follow the IA32e layout as given below:



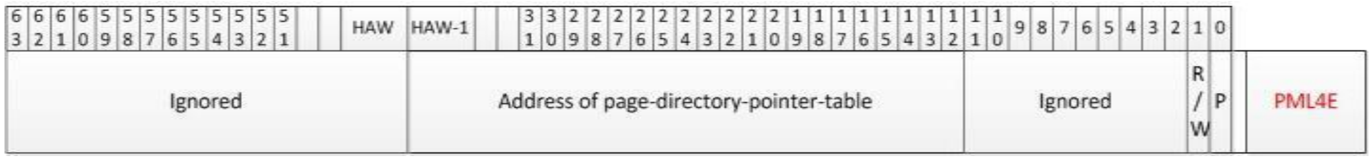
63 62 61 60 59 58 57 56 55 54 53 52 51 … HAW	HAW-1 … 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0												
Ignored	Address of page-directory-pointer-table	Ignored										R/W	P	PML4E
Ignored	Address of the 1GB page	Reserved	PAT	Ign	Null	Ign	1	Ign	PCD	PWT	Ign	R/W	P	PDPE (1GB Page)
Ignored	Address of page-directory-table	Ignored										R/W	P	PDPE Page Directory
Ignored	Address of the 2MB page	Reserved	PAT	Ign	Null	Ign	1	Ign	PCD	PWT	Ign	R/W	P	PDE (2MB Page)
Ignored	Address of page-table	Ignored										R/W	P	PDE Page Table
Ignored	Address of the 64KB MB page	Rsvd.	Ign	Null	Ign	PAT	Ign	PCD	PWT	Ign		R/W	P	PTE (64KB Page)
Ignored	Address of 4KB Page	Ign	Null	Ign	PAT	Ign	PCD	PWT	Ign			R/W	P	PTE (4KB Page)

Each table entry is further broken down along with the required functions. GFX has a 4 level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required for support.

Pointer to PML4 table

In legacy mode, pointer to PML4 table is provided via the context descriptor.

PML4E: Pointer to PDP Table



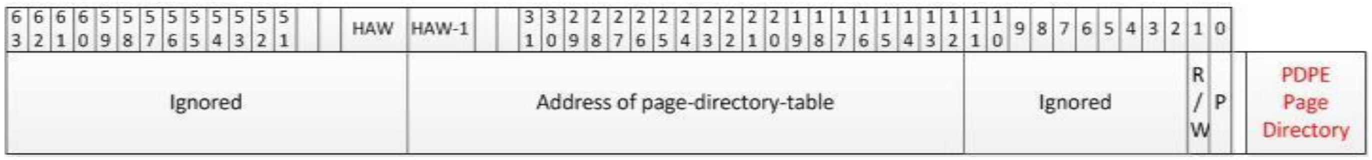
Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PML4 Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE: Pointer to PD Table

PDP entry is used to locate the page directory. IA32e supports 1GB pages, the PDPE has a mechanism to identify a way to say whether this PDPE represents a pointer to page directory or to a contiguous 1GB physical memory.

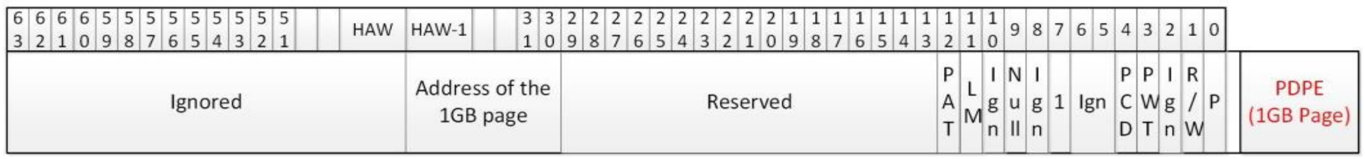
PDPE for PD



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. Access rights are described later. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. It must be “1” to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE for 1GB Page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):30	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
29:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS. For other configurations it is ignored
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	This bit must be "1" to point to a valid Page.

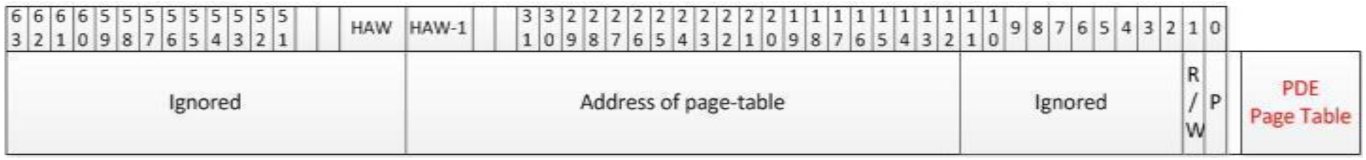
* HAW = 39 for client, and 46 for server.

PD: Pointer to Page Table

This section describes the following:

- PDE for Page Table
- PDE for 2 MB Page

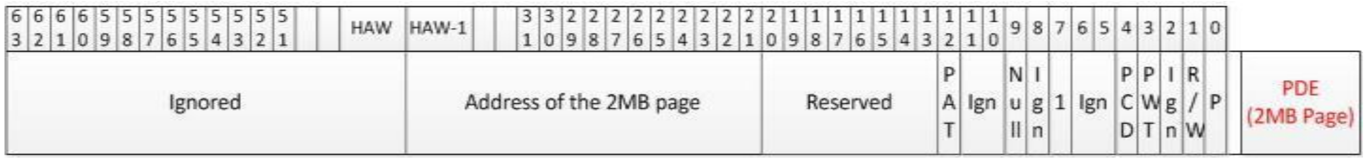
PDE for Page Table



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page- table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. The value must be "1" to point to a page directory pointer table.

* HAW = 39 for client, and 46 for server.

PDE for 2MB Page

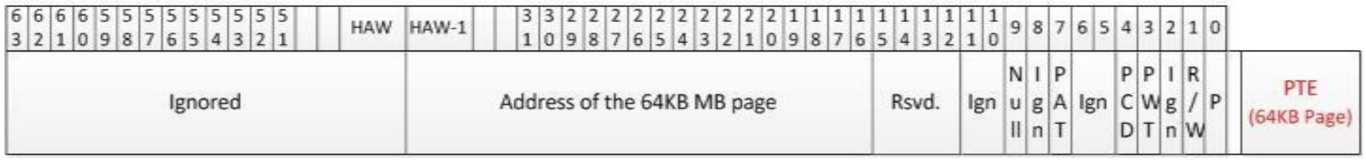


Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):21	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
20:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS. For other configurations it is ignored
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*

Bits	Field	Description
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 1GB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 64KB Page

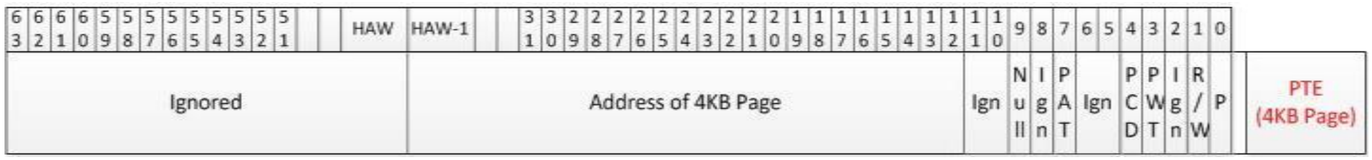


Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS.
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*

Bits	Field	Description
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be “1” to point to a 64KB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 4KB Page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 64KB memory page referenced by this entry.This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry) to the memory region controlled by this entry. See a later section for access rights. GPU does not support Supervisor mode contexts.
0	P: Present	It must be "1" to point to a 4KB Page.

* HAW = 39 for client, and 46 for server.

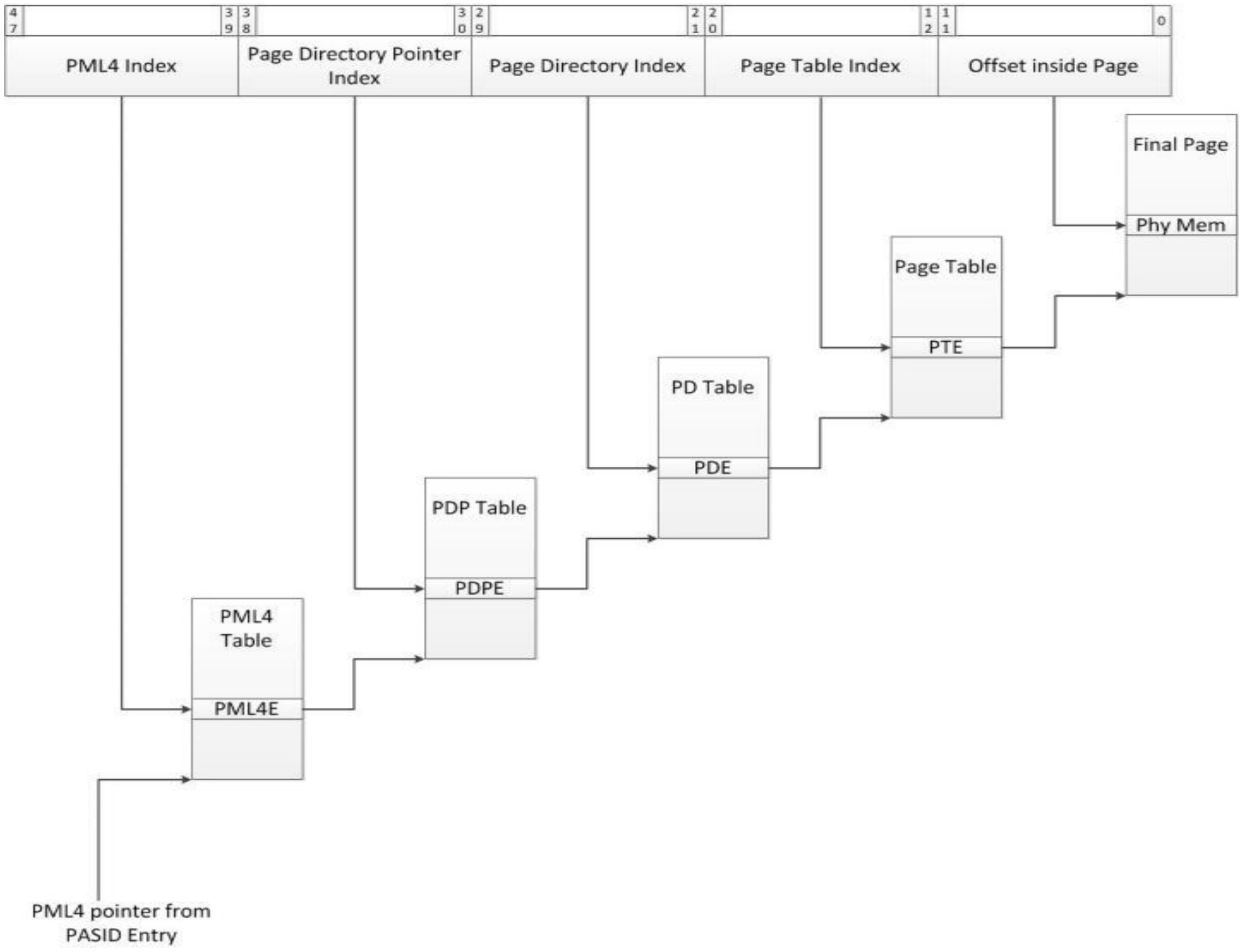
Advanced mode with 48b VA and IA32e Support

In advanced mode, Gen9 per process GTT mechanism supports IA32e compatible page tables. Paging mechanism is controlled via IOMMU which shall be owned by OS or GFX driver (not both at the same time).

Page Walk in Advanced Mode

For page walk in advanced mode with 48b VA, we need 4 levels. The walk will start with a PML4 table pointer extracted from PASID entry and uses the 48b VA as index to consecutive levels of page tables.

The following diagram shows the page walk that is needed for a 4KB page.



64bit (48b canonical) address requires 4-levels of page table format where the context carries a pointer to highest level page table (PML4 pointer) via PASID. The rest of the walk is normal page walk thru various levels.

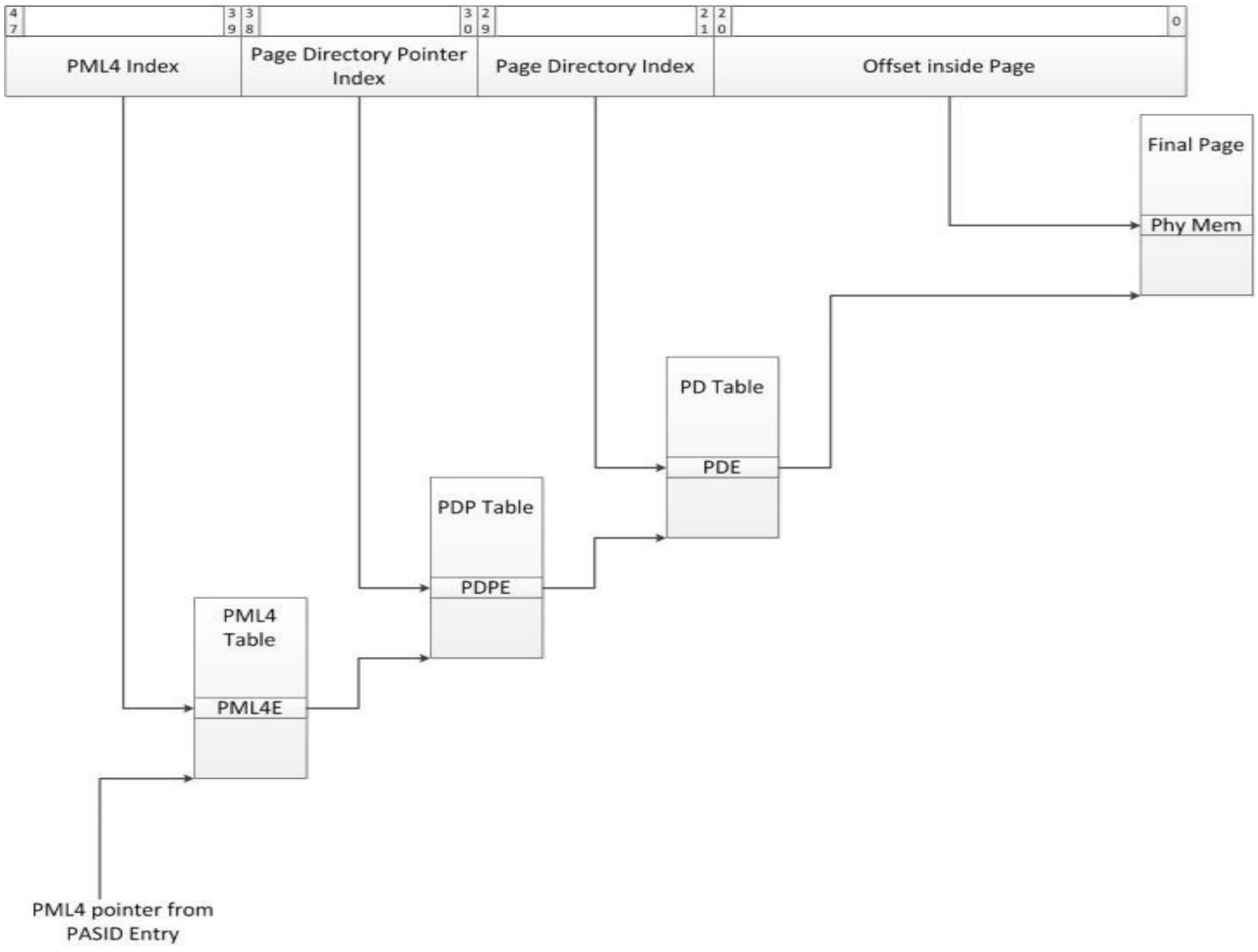
To repurpose the caches the following mechanism will be used:

- 3d: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache
- Media: 4KB to store PML4, 4KB as PDP cache, 2x4PD cache
- VEBOX, Blitter: each with a 4KB acting as PML4, PDP, PD cache.

Note: design can section the 512 entries within 4KB to separate areas for PML4, PDP and PD.

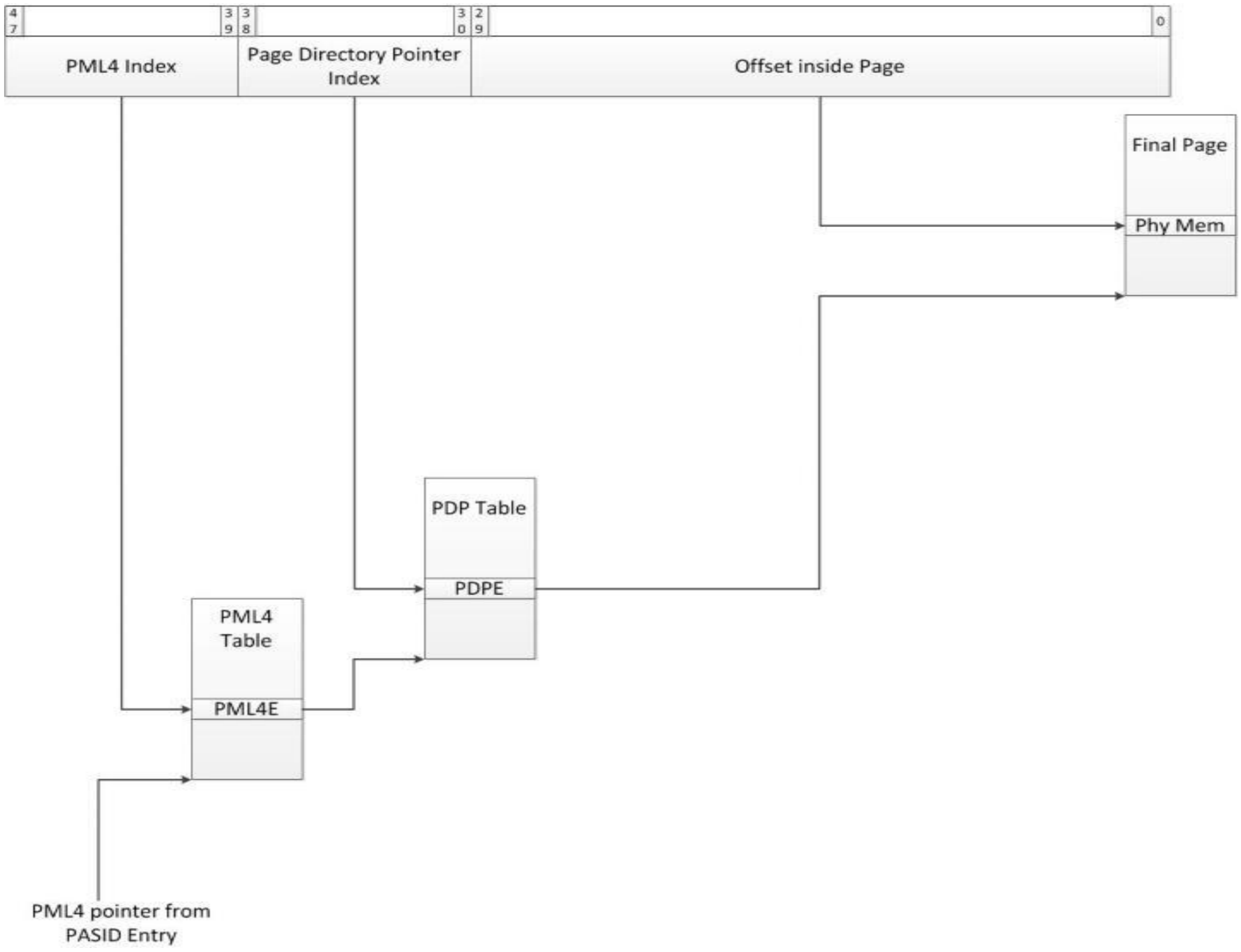
Walk with 2MB Page

With the 2MB Page walk, last level of the page walk is skipped where the PD entry points to the final page.



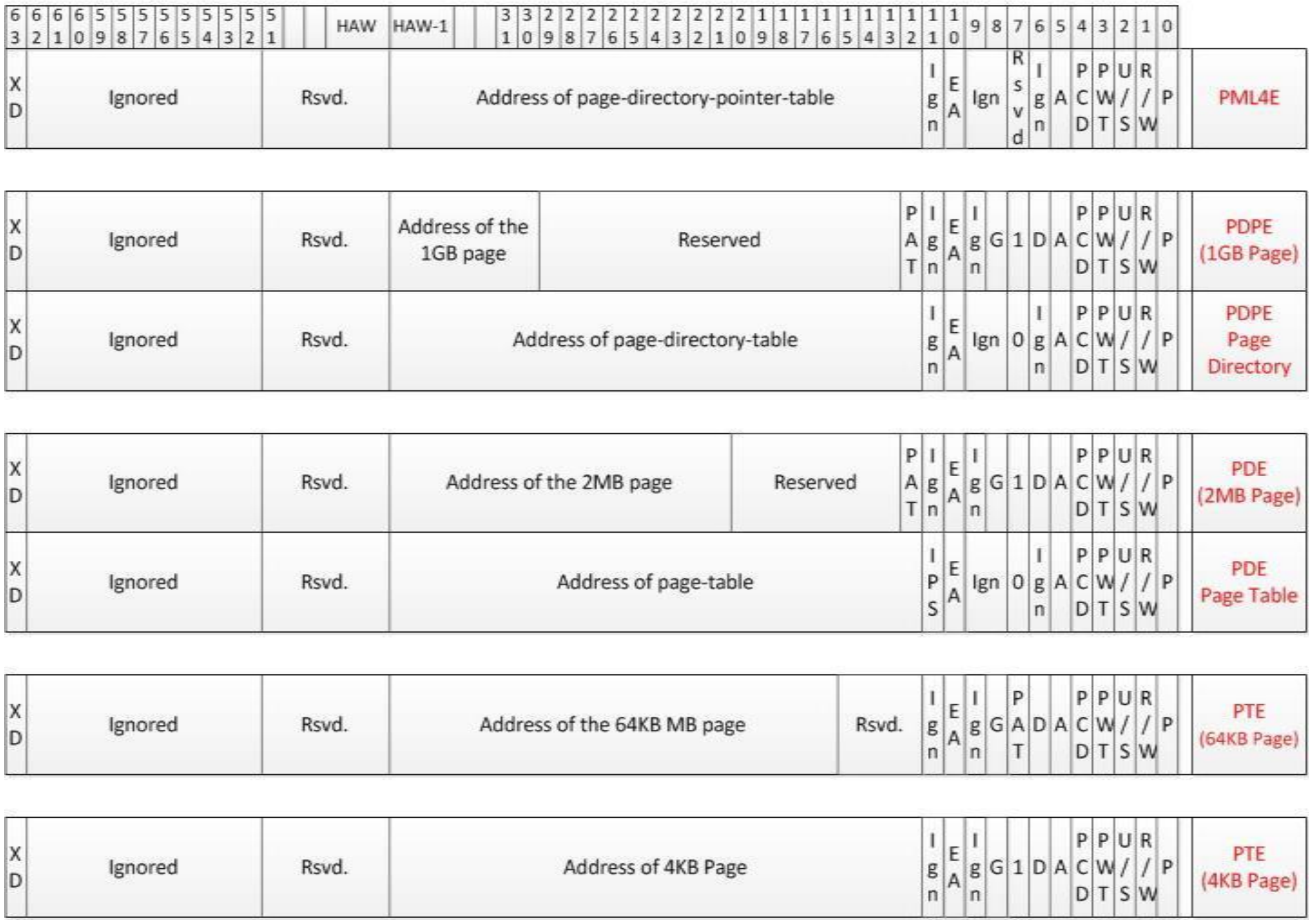
Walk with 1GB Page

For the support for 1GB page size, the following mechanism is needed.



Page Tables Entry (PTE) Formats

Page Table Entry (PTE) formats will follow the IA32e layout as given below:



63	62	61	60	59	58	57	56	55	54	53	52	51		HAW	HAW-1		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

XD	Ignored	Rsvd.	Address of page-directory-pointer-table	Ign	EA	Ign	Rsvd	Ign	A	PCD	PWT	U/S	R/W	P	PML4E

XD	Ignored	Rsvd.	Address of the 1GB page	Reserved	PAT	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P	PDPE (1GB Page)
XD	Ignored	Rsvd.	Address of page-directory-table			Ign	EA	Ign	0	Ign	A	PCD	PWT	U/S	R/W	P		PDPE Page Directory

XD	Ignored	Rsvd.	Address of the 2MB page	Reserved	PAT	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P	PDE (2MB Page)
XD	Ignored	Rsvd.	Address of page-table			IPS	EA	Ign	0	Ign	A	PCD	PWT	U/S	R/W	P		PDE Page Table

XD	Ignored	Rsvd.	Address of the 64KB MB page	Rsvd.	Ign	EA	Ign	G	PAT	D	A	PCD	PWT	U/S	R/W	P	PTE (64KB Page)

XD	Ignored	Rsvd.	Address of 4KB Page	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P	PTE (4KB Page)

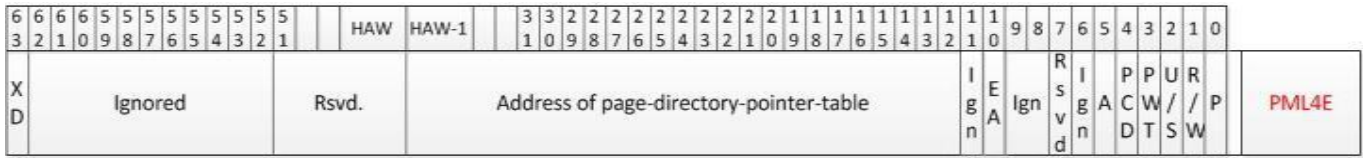
Each table entry is further broken down along with the required functions. GFX has a 4 level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required for support.

Pointer to PML4 table

Page table pointer is the starting address where the PML4 table starts. The contents of pointer will be provided by PASID table entry in case of advanced context, else it will be provided by software as part of the legacy context with 48b addressing.

Details of PASID entry is given in later sections.

PML4E: Pointer to PDP Table



Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. Not support in gen9
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9:8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	*Reserved*	*Reserved (must return 0's)*
6	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. GPU does not support any memory type but WB when accessing paging structures.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry. GPU does not support any memory type but WB when accessing paging

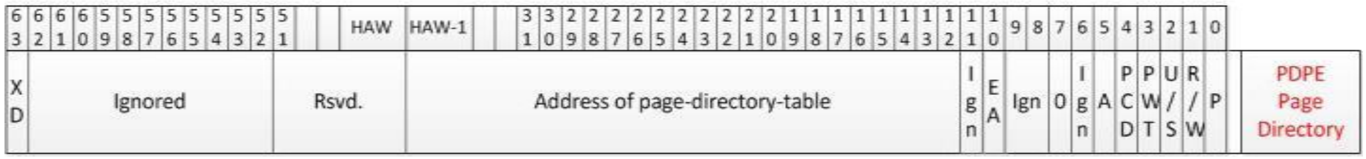
Bits	Field	Description
		structures.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PML4 Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE: Pointer to PD Table

PDP entry is used to locate the page directory. IA32e supports 1GB pages, the PDPE has a mechanism to identify a way to say whether this PDPE represents a pointer to page directory or to a contiguous 1GB physical memory.

PDPE for PD

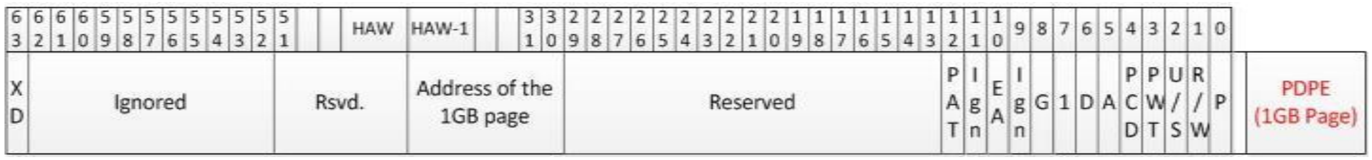


Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. Not support in gen9
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended	Extended Access bit is added for devices to separate accesses from IA cores. If

Bits	Field	Description
	Access	EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9:8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	*Reserved*	*Reserved (must return 0's)*
6	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. GPU does not support any memory type but WB when accessing paging structures.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry. GPU does not support any memory type but WB when accessing paging structures.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE for 1GB Page



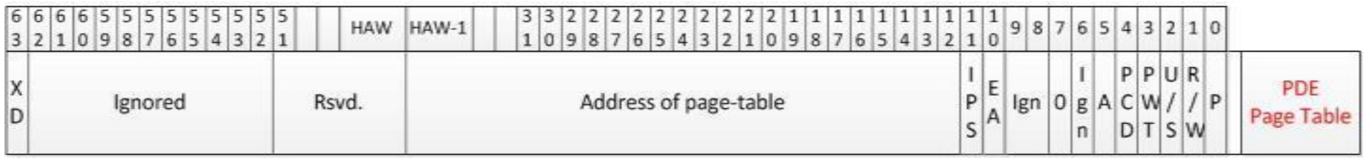
Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. Not support in gen9
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):30	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
29:13	Reserved	*Reserved (must return 0's)*
12	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	Page Size	Must be 1 to indicate 1GB page.
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.

Bits	Field	Description
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	The value must be "1" to point to a 1GB Page.

* HAW = 39 for client, and 46 for server.

PD: Pointer to Page Table

PDE for Page Table

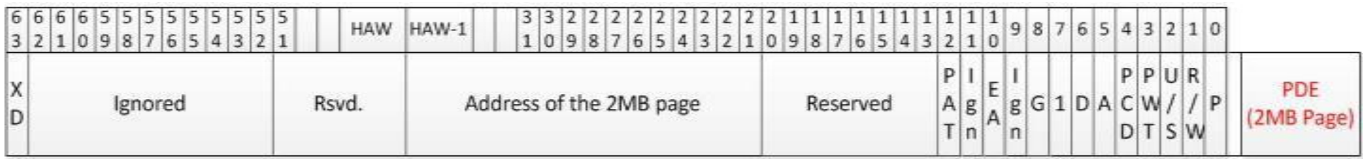


Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. Not support in gen9
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	IPS	If FL64KPE=1 in the corresponding PASID entry, the page table referenced by this PD entry with IPS=1 translates into 64KB pages. If IPS=0, the page table

Bits	Field	Description
		referenced here translates into 4KB pages. If FL64KPE=0 in the corresponding PASID entry, the IPS value is ignored and the page table referenced by this entry translates into 4KB pages.
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9:8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	*Reserved*	*Reserved (must return 0's)*
6	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. GPU does not support any memory type but WB when accessing paging structures.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry. GPU does not support any memory type but WB when accessing paging structures.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PD Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDE for 2MB Page

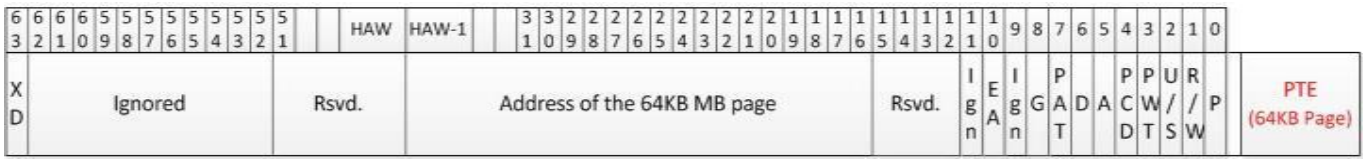


Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. Not support in gen9
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):21	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
20:13	Reserved	*Reserved (must return 0's)*
12	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
11	Ignored/Reserved	Ignored/not used by hardware
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	Page Size	Must be 1 to indicate 2MB page.
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.

Bits	Field	Description
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 2MB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 64KB Page

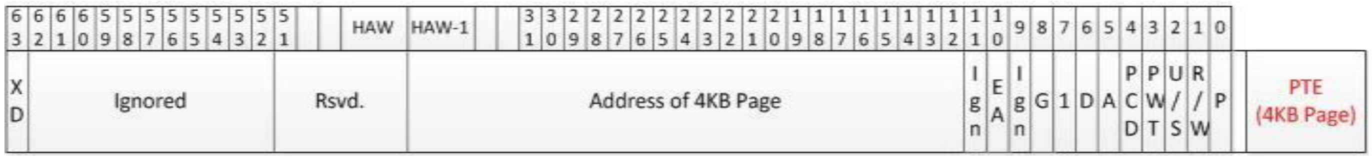


Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. Not support in gen9
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:12	*Reserved*	*Reserved (must return 0's)*
11	*Ignored/Reserved*	Ignored/not used by hardware
10	EA: Extended	Extended Access bit is added for devices to separate accesses from IA cores. If

Bits	Field	Description
	Access	EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 64KB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 4KB Page



Bits	Field	Description
63	XD: Execute Disable	If NXE=1 in the relevant extended-context-entry, execute permission is not granted for requests to the memory region controlled by this entry when XD=1. Not support in gen9
62:52	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW*	*Reserved*	*Reserved (must return 0's)*
(HAW-1):12	ADDR: Address	Physical address of 4KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11	*Ignored/Reserved*	*Ignored/not used by hardware*
10	EA: Extended Access	Extended Access bit is added for devices to separate accesses from IA cores. If EAFE=1 in the relevant PASID-entry, this bit indicates whether this entry has been used for address translation by device. It is the devices responsibility to set this bit. If EAFE=0 in the relevant PASID-entry, this bit is ignored. This bit applies to GPU Only.
9	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
8	G: Global	If PGE=1 in the corresponding context table entry, this field can be set by s/w to indicate that the memory region pointed by this entry can be considered global *Global paging is not used by GPU.*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6	D: Dirty	D-bit needs to be managed by h/w as the table entry is accessed with a successful write transaction. See later sections for A/D-bit management.
5	A: Accessed	A-bit needs to be managed as the table entry being accessed. Hardware needs to set this bit for the first access to the region defined with this page table entry. See later sections for A/D-bit management.
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.

Bits	Field	Description
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	U/S: User/Supervisor	User vs supervisor access rights. If 0, requests with user-level privilege are not allowed to the memory region controlled by this entry. See section for access rights. *GPU does not support Supervisor mode contexts.*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 4KB Page.

* HAW = 39 for client, and 46 for server.

GTT Cache

Processor graphics page walker implements a GTT cache which holds the remaining entries that are read as a cacheline but not used for the immediate page walk. This is only applicable in case of leaf walks and not including the 2MB/1GB page sizes. When SW enables the use of 2MB/1GB page sizes, it must disable the GTT cache.

When running advanced context, GTT cache must be disabled. This is needed as the GTT cache is not snooped, and no A/D bit updates are done on the PTEs that are cached in the GTT Cache.

GFX Page Walker (GAM)

GPU supports various engines behind the same page walker. These streams/contexts are identified Client level IDs which are carried via the arbitration pipeline. Page walker using look-up tables does the correct selection for the page tables in case of concurrent context are running at the same time.

There are two different types of page table types:

Global graphics translation table (GGTT) is a single common translation table used for all processes. There can be many Per-process graphics translation table (PPGTT). This requires an additional lookup for translation.

Virtual Memory Structure	Memory Location
Global (GGTT)	GSM Only
Per-Process (PPGTT) – private	2 to4-level, Page Tables anywhere

Virtual Memory Structure	Memory Location
Per-Process (IA32e) – shared	4 levels, Page Tables anywhere

IA32e compatible PPGTT is added to gen8/gen9 to enable SVM (shared virtual memory) functions.

Context Definition for GFX Page Walker

Page Walker blocks need details about the context to decide on what type of page tables are used, what the error handling cases are, and many other details to operate. The information is passed to Page Walker (GAM) by the respective command streamer/DMA.

GAM needs to support the following engines:

- Render
- Media (VDBox) x2
- Blit
- VEBOX x2
- WDBox

The following fields are sent to GAM:

- **Context Type** (4 bits):
  - **Legacy vs Advanced Context.** Defines the context type and qualifies the rest of the fields. Same field may mean something else between the *Legacy* vs *Advanced* context. There is no restriction for what type of context can run in either combination.
    - *Requests without address-space-identifier (Legacy Context):* These are the normal memory requests from endpoint devices. These requests typically specify the type of access (read/write/atomics), targeted DMA address/size, and identity of the device originating the request.
    - *Requests with address-space-identifier (Advanced Context):* These are memory requests with added information identifying the targeted process address space from endpoint devices supporting virtual memory capabilities. Beyond attributes in normal requests, these requests specify the targeted process address space identifier (PASID), and extended attributes such as Execute-Requested (ER) flag (to indicate reads that are instruction fetches), and Privileged-mode-Requested (PR) flag (to distinguish user versus supervisor access). For details, refer to the Process Address Space ID (PASID) Capability in the PCI-Express specifications.
  - **A/D Support Enable.** Access and Dirty bits are used when OS is managing the page tables and has been added to IA32e compatible page walk. Context defines whether A/D bits need to be managed via GPU (only applicable in Advanced Context).
  - **Privileged Context Support.** Enables GPU to be able to run a privileged context which translates into page table accesses regardless of user vs supervisor privileges (only applicable in Advanced Context).

    - **32b vs 48b VA Support.** Enables 48b VA in page tables for the page walks. The rest of the HW is seamless to 32b vs 48b VA address walks, however GAM does the check and properly aligns the page walk to address bits. **Note:** Only applicable in Legacy Context. Advanced Context is always 48b.
    - **Page Fault Support Model:**
        - *Fault and Hang:* The only supported fault handling mode for legacy context and it is not applicable to advanced mode. Optionally hang can be skipped for HW to make progress (same as Gen7.5).
        - *Fault and Stream (Switch if needed):* Context can survive thru a number of page faults and could be switched out by the scheduler if a certain threshold is reached.
        - *Fault and Halt:* HW detects page fault and reports to SW; the request is flagged in pending queue as "waiting for page response" and is halted until the page response is returned.
- **Function Number** – 3-bit field that defines the function number of the device. GFX device is always on BUS=0 and DEVICE=2. If we are not virtualized, our FUNCTION#=0 however if virtualized function number can be any 8 possible values (i.e. 0-7). The BUS/DEVICE/FUNCTION numbers are used for the initial walk for ROOT and CONTEXT tables.
- **PASID** – Process Address Space IDentifier: Use to identify the context that is submitted to HW. We use the PASID in many places where during the page walk (i.e. PASID table look up) or while communicating with SW on page faults. Each engine could be running an independent context with different PASID. The page walker should have a mechanism to be able to cache at least some number of PASID table entries (matching the engine count) for faster walk.
- **Context ID** (Queue ID, Bell ID) – Context ID is used to further qualify the running context beyond the PASID. PASID is given per process, and same process may allocate multiple queues to communicate with HW. The only way to further identify the process is to use an additional ID. For GFX HW Context ID could be same as the bell number assigned to it. GAM HW uses the context ID to populate the queue ID field while communicating page faults to SW.
- **Page Table Pointers** – The field could be up to 256 bits (i.e. 4x64bits) to identify the page table pointers associated with the context. For legacy 32b context, the entire 256b is valid representing the 4 PDPTR table entries. For 48b legacy context only the lower 64b is relevant pointing to base of PML4. In case of advanced context, PASID is given in the context definition.

Context Definition Delivery

Context Definition is supposed to be delivered from the corresponding command streamer to GAM and GAM has independent storage for each engine present. WDBOX is an exception because it does not have a command streamer; its context definition is default.

Context Definition is given by *CS to GAM via a new message:

Message: "Context Available"

GAM prepares for new context, cleans up internal state and does the proper fencing. Most of these steps should have been performed when context switch request was done for the previous context, but added here for completeness.

Message: "Context Receive Ready"

GAM is ready for the context. *CS writes all new context values into the descriptor registers. To push all context descriptors CS sends the following message to GAM also indicating new context descriptor is downloaded.

Message: "Context Launched"

GAM does the context requirements and sends the following message to CS to resume its command parser.

Message: Context Confirmed

GAM should send context confirmed message only after PD restore is done. CS waiting for context confirmed message is treated as PD restore busy. Since all clients memory interface are blocked during PD restore it doesn't make any difference if the context confirmed message is send by GAM immediately or after PD restore.

Element Descriptor Register

General Description	Element Information: The register is populated by command streamer and consumed by GAM
Register Offset	See per engine list below.

Bits	Access	Default	Field
63:32	RO	Xh	**Context ID:** Context identification number assigned to separate this context from others. Context IDs needs to be recycled in such a way that there cannot be two active contexts with the same ID. This is a unique identification number by which a context is identified and referenced.
31:12	RO	Xh	**LRCA:** Command Streamer Only
11:9	RO	Xh	**Function Number:** GFX device is considered to be on Bus0 with device number of 2. Function number is normally assigned as 000b. Not used in Gen8/9.
8	RO	Xh	**Privileged Context / GGTT vs PPGTT mode:** Differs in legacy vs advanced context modes: **In Legacy Context:** Defines the page tables to be used. This is how page walker come to know PPGTT vs GGTT selection for the entire context. 0: Use Global GTT 1: Use Per-Process GTT

Bits	Access	Default	Field
7:6	RO	Xh	**Fault Model:** 00b: Fault & Hang. Same mode as Gen7.5.
5	RO	Xh	**Deeper IA coherency Support:** **In Advanced Context:** Defines the level of IA coherency: 0: IA coherency is provided at LLC level for all streams of GPU (i.e. Gen7.5 like mode). 1: IA coherency is provided at L3 level for EU data accesses of GPU.
4	RO	Xh	**A&D Support / 32&64b Address Support:** Differs in legacy vs advanced context modes: **In Legacy Context:** Defines 32b vs 64b (48b canonical) addressing format: 0: 32b addressing format. 1: 64b (48b canonical) addressing format. **In Advanced Context:** Defines A&D bit support: 0: A&D bit management in page tables is NOT supported. 1: A&D bit management in page tables is supported.
3	RO	Xh	**Context Type: Legacy vs Advanced** Defines the context type. 0: Advanced Context: Defines the rest of the advanced capabilities (i.e. OS page table support, fault models, ...). Note that advanced context is not bounded to GPGPU. 1: Legacy Context: Defines the context as legacy mode which is similar to prior generations of Gen8. **Note:** Bits [8:4] differs in functions when legacy vs advanced context modes are selected.
2	RO	Xh	**FR:** Command streamer specific.
1	RO	Xh	Always '1'
0	RO	Xh	**Valid:** Indicates that element descriptor is valid. If GAM is programmed with an invalid descriptor, it continues but flags an error.

PDP0/PML4/PASID Descriptor Register

General Description	PDP0/PML4/PASID: The register is populated by command streamer and consumed by GAM. It contains one of the 3 values which is determined by looking at the element descriptor.
Register Offset	See per engine list below

Bits	Access	Default	Field
63:0	RO	Xh	**PDP0/PML4/PASID:** This register can contain three values which depend on the element descriptor definition. **PASID[19:0]**: Populated in the first 20bits of the register and selected when Advanced Context flag is set. **PML4[38:12]:** Pointer to base address of PML4 and selected when Legacy Context flag is set

Bits	Access	Default	Field
			and 64b address support is selected **PDP0[38:12]:** Pointer to one of the four page directory pointer (lowest) and defines the first 0-1GB of memory mapping *Note: This is a guest physical address*

PDP1 Descriptor Register

General Description	PDP1: The register is populated by command streamer and consumed by GAM. It contains one of the pointers to PD.
Register Offset	See per engine list below

Bits	Access	Default	Field
63:12	RO	Xh	**PDP1:** Pointer to one of the four page directory pointer (lowest+1) and defines the first 1-2GB of memory mapping *Note: This is a guest physical address*

PDP2 Descriptor Register

General Description	PDP2: The register is populated by command streamer and consumed by GAM. It contains one of the pointers to PD.
Register Offset	See per engine list below

Bits	Access	Default	Field
63:12	RO	Xh	**PDP2:** Pointer to one of the four page directory pointer (lowest+2) and defines the first 2-3GB of memory mapping *Note: This is a guest physical address*

PDP3 Descriptor Register

General Description	PDP3: The register is populated by command streamer and consumed by GAM. It contains one of the pointers to PD.
Register Offset	See per engine list below

Bits	Access	Default	Field
63:12	RO	Xh	**PDP3:** Pointer to one of the four page directory pointer (lowest+3) and defines the first 3-4GB of memory mapping *Note: This is a guest physical address*

List of Registers and Command Streamers

The following registers are message registers and not written directly by SW.

Engine	Offset	Description
Render	x4400h	Element Descriptor Register
	x4408h	PDP0/PML4/PASID Descriptor Register
	x4410h	PDP1 Descriptor Register
	x4418h	PDP2 Descriptor Register
	x4420h	PDP3 Descriptor Register
Media0 (VDBOX0)	x4440h	Element Descriptor Register
	x4448h	PDP0/PML4/PASID Descriptor Register
	x4450h	PDP1 Descriptor Register
	x4458h	PDP2 Descriptor Register
	x4460h	PDP3 Descriptor Register
Media1 (VDBOX1)	x4480h	Element Descriptor Register
	x4488h	PDP0/PML4/PASID Descriptor Register
	x4490h	PDP1 Descriptor Register
	x4498h	PDP2 Descriptor Register
	x44A0h	PDP3 Descriptor Register
VEBOX	x44C0h	Element Descriptor Register
	x44C8h	PDP0/PML4/PASID Descriptor Register
	x44D0h	PDP1 Descriptor Register
	x44D8h	PDP2 Descriptor Register

Engine	Offset	Description
	x44E0h	PDP3 Descriptor Register
Blitter	x4500h	Element Descriptor Register
	x4508h	PDP0/PML4/PASID Descriptor Register
	x4510h	PDP1 Descriptor Register
	x4518h	PDP2 Descriptor Register
	x4520h	PDP3 Descriptor Register

Messages:

Message Name	Source	Destination	Category	Address	Bit	Mask Bit	Value	Description
Context Available	CS (GT)	GAM (GT)	self-clear	4004	0	16	1	Signal request from CS to GAM as new context is about to be submitted.
Context Receive Ready	GAM (GT)	CS(GT)	self-clear	3438	0	16	1	Signal ack from GAM to CS in response to Context Available message from CS to GAM.
Context Launched	CS (GT)	GAM (GT)	self-clear	4004	1	17	1	Signal indicator to GAM that context descriptor is pushed.
Context Confirmed	GAM (GT)	CS(GT)	self-clear	3438	1	17	1	Signal ack from GAM to CS in response to Context Launched message from CS to GAM.
Context Available	BCS (GT)	GAM (GT)	self-clear	4014	0	16	1	Signal request from CS to GAM as new context is about to be submitted.
Context Receive Ready	GAM (GT)	BCS(GT)	self-clear	23438	0	16	1	Signal ack from GAM to BCS in response to Context Available message from BCS to GAM.
Context Launched	BCS (GT)	GAM (GT)	self-clear	4014	1	17	1	Signal indicator to GAM that context descriptor is pushed.
Context Confirmed	GAM (GT)	BCS(GT)	self-clear	23438	1	17	1	Signal ack from GAM to BCS in response to Context Launched message from BCS to GAM.
Context Available	VECS (GT)	GAM (GT)	self-clear	4010	0	16	1	Signal request from CS to GAM as new context is about to be submitted.
Context Receive Ready	GAM (GT)	VECS(GT)	self-clear	1B438	0	16	1	Signal ack from GAM to VECS in response to Context Available message from VECS to GAM.

Message Name	Source	Destination	Category	Address	Bit	Mask Bit	Value	Description
Context Launched	VECS (GT)	GAM (GT)	self-clear	4010	1	17	1	Signal indicator to GAM that context descriptor is pushed.
Context Confirmed	GAM (GT)	VECS(GT)	self-clear	1B438	1	17	1	Signal ack from GAM to VECS in response to Context Launched message from VECS to GAM.
Context Available	VCS0 (GT)	GAM (GT)	self-clear	4008	0	16	1	Signal request from CS to GAM as new context is about to be submitted.
Context Receive Ready	GAM (GT)	VCS0(GT)	self-clear	13438	0	16	1	Signal ack from GAM to VCS in response to Context Available message from VCS to GAM.
Context Launched	VCS0 (GT)	GAM (GT)	self-clear	4008	1	17	1	Signal indicator to GAM that context descriptor is pushed.
Context Confirmed	GAM (GT)	VCS0(GT)	self-clear	13438	1	17	1	Signal ack from GAM to VCS in response to Context Launched message from VCS to GAM.
Context Available	VCS1 (GT)	GAM (GT)	self-clear	400C	0	16	1	Signal request from CS to GAM as new context is about to be submitted.
Context Receive Ready	GAM (GT)	VCS1(GT)	self-clear	1D438	0	16	1	Signal ack from GAM to VCS in response to Context Available message from VCS to GAM.
Context Launched	VCS1 (GT)	GAM (GT)	self-clear	400C	1	17	1	Signal indicator to GAM that context descriptor is pushed.
Context Confirmed	GAM (GT)	VCS1(GT)	self-clear	1D438	1	17	1	Signal ack from GAM to VCS in response to Context Launched message from VCS to GAM.

Updating Page Table Pointers (aka PD Load)

In case of legacy context, driver is allowed to add/remove pages as long as it is ensured that h/w is not using these entries. Pre-gen8 flow allowed a mid-context PD load to update the PD entries and directed h/w to reload updated entries.

Pre-loading of Page Directory Entries (PD load) for 32b legacy mode is not supported from Gen9 onwards. PD entries are loaded on demand when there is a miss in the PDE cache of the corresponding page walker. Any new page additions by the driver are transparent to the HW, and the new page translations will be fetched on demand. However, any removal of the pages by the driver should initiate a TLB invalidation to remove the stale entries.

Page Walker (GAM) Reset

GAM gets all the engine specific resets as well as device and bus resets to manage its internal logic domains. It is the expectation of SW when a particular GPU engine (i.e. Render, Media...) gets reset, all its related HW is cleared and comes out fresh for reprogramming. That is true for most of the logic with the exception of some shared HW blocks. The following blocks require additional steps (post-reset) from SW to further clean-up the HW:

- **Hardware TLBs**: The caching structures for the page walks are often considered shared resources. The expectation for GFX driver to clear the TLBs via "TLB Invalidate" prior to re-using the engine post reset. This is the same process that was followed on previous GPU generations.
- **Page Requests**: At the time of the reset HW may have outstanding page requests to SW for page faulted accesses. These requests could be at any level hence it is required for SW to clear these paging requests pre/post-engine reset. Engine reset ensures that no new page requests are sent from HW. Page requests could be at the "page request queue" in memory where they could be mapped to a dummy page post engine reset completion. Or they could be at the MMIO registers which will block completion of the reset; it is up to SW to service paging request interrupts without waiting for the completion of reset request.

Device reset (FLR) covers most of the page walker. However, there are exceptions where all messaging towards the rest of the system (system agent) should not be impacted by it.

All external interactions and IOMMU related blocks are kept under bus (system) reset. GAM keeps the following blocks outside the device reset:

- IOMMU registers and content
- All system agent messaging structures (including translation enable flows, root pointer structures, and DMA fault reporting pieces)

An engine being reset also means the particular context that engine is running, is complete or taken out. This requires GAM to decrement the PASID_State Counter if the engine was running a PASID based (advanced) context. For FLR (device reset) similar requirement holds. In case of device reset, GAM needs to decrement all the PASID state counters that are active on the GPU before completing the sequence.

TLB Caching and Management

As compared to previous generation of TLB entry, IA32e page translation entry is quite different. At every stage of the page different bits need to be taken into account and proper treatment is required. Regardless of PPGTT vs GGTT usage, the paging entry has the same format. Linear address are translated using a hierarchy of in-memory paging structures located using the contents of CR3. IA-32e paging translates 48-bit linear addresses to 52-bit physical addresses.1 Although 52 bits corresponds to 4 PBytes, linear addresses are limited to 48 bits; at most 256 TBytes of linear-address space may be accessed at any given time.IA-32e paging may map linear addresses to 4-KByte pages, 2-MByte pages, or 1-GByte pages.

63	62 … 52	51 … M[1]	M-1 … 30	29 … 21	20 … 13	12	11 10 9	8	7	6	5	4	3	2	1	0	
Reserved[2]			Address of PML4 table				Ignored					PCD	PWT	Ign.			CR3
XD3	Ignored	Rsvd.	Address of page-directory-pointer table				Ign.		Rsvd	Ign	A	PCD	PWT	U/S	R/W	1	PML4E: present
Ignored																0	PML4E: not present
XD	Ignored	Rsvd.	Address of 1GB page frame	Reserved		PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1	PDPTE: 1GB page
XD	Ignored	Rsvd.	Address of page directory				Ign.		0	Ign	A	PCD	PWT	U/S	R/W	1	PDPTE: page directory
Ignored																0	PDTPE: not present
XD	Ignored	Rsvd.	Address of 2MB page frame		Reserved	PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 2MB page
XD	Ignored	Rsvd.	Address of page table				Ign.		0	Ign	A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored																0	PDE: not present
XD	Ignored	Rsvd.	Address of 4KB page frame				Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored																0	PTE: not present

The following rules apply:

1. *M is an abbreviation for MAXPHYSICAL ADDRESS*
2. *Reserved fields must be "0"*
3. *Ignored field must be ignored (there could be private information)*
4. *All ignore options are part of the context entry and coming from IOMMU definition.*

TLB Caches

For gen8/9 the caching structures are separated as following with the architectural view, this is also applicable to s/w view of these caches when it comes to invalidations.

Context Cache - CC

This is the storage for context table entry which is achieved as part of root/context table walk.

Context cache can also be invalidated with directed invalidations, where HW needs to invalidate the content of the context cache along with all low level caches.

PASID Cache - PC

This is where the HW copy of the PASID table entry is kept and it is per context. This makes it unique for every HW engine that could be running an independent context (per GAM):

- Render/GPGPU
- MFX (VDBOX) – 1
- MFX (VDBOX) – 2
- Video Enhancement (VEBOX) – 1
- Video Enhancement (VEBOX) – 2
- Blitter

The cache content is updated if the corresponding engine is running an advanced context where its page table pointers are accessible via PASID table. In case of legacy context running engine, corresponding PASID Cache entry is not valid. Recommendation is to keep ONE physical storage per engine which is filled/invalidated during the context switch time.

PASID Cache can also be invalidated with the directed invalidations along with low level caches and needs to be re-filled prior to context resuming.

Intermediate Page Walk Caches (PML4, PDP, PD) – PWC

These are the stages where intermediate page walk entries are cached to speed-up/shorten the page walk when final TLB is missed. Each level can be cached separately or along with different levels, the cacheability structures will have programmability to move the boundary of different levels to accommodate more/less on each page walk level. However, as a concept, for legacy 32b addressing mode, requirement is to cache 4PDPs along with 4x4KB PDs for certain engines, at least for render and media. The others will use cache concept.

TLB – Final Page Entry

The size of the TLBs has been increased over the previous generation and should be targeting using the following list:

- L3 TLB: 768 TLB entries – This is where all HDC, I$, Constant, State, and Sampler streams are stored.

- MFX: 512 TLB entries – All Media streams (split 256/256 between two media engines).
- BLT: 32 entries.
- Z: 512 TLB entries – All depth accesses.
- C: 256 (256 TLB entries) – All color accesses.
- FF: 128 (128 TLB entries) – All FF accesses to memory.
- VLF: 32 (32 TLB entries) – Media surface.
- GAV: 64 (64 TLB entries) – Video enhancement.
- WiDi: 64 (64 TLB entries) – Wireless Display.

All TLB entries are increased to 48b to contain larger address as well as the page attributes attached to it.

The max size of a single TLB is 256 entries, larger quantities have to be handled as set-associative storages. Set associativity will be managed by low order page bits (i.e. address#12, address#13, ...).

Both Color and Z TLBs are designed to process a single memory request per cycle. To achieve a higher throughput where concurrent Color or Z read/write's are used, following register bit needs to be enabled: mmio0x04A30h [31]

The sizes of RCCTLB and ZTLB is different in SKL. In SKL both these have 448 entries.

The size of the L3 TLB is also different between projects. The default TLB entry allocations are:

- SKL (L3TLB-Gfx **640**): L3(**80**:*0-79*), DC(**100**:*80-179*), TX(**444**:*180-623*), GATR(**16**:*624-639*)
- SKL (L3TLB-GPGPU **640**): L3(**80**:*0-79*), DC(**460**:*80-539*), TX(**100**:*540-639*)

For giving more TLB resources for both DC and TX, the following allocations are recommended.

- SKL (L3TLB-Gfx **640**): L3(**80**:*0-79*), DC(**544**:*80-623*), TX(**544**:*80-623*), GATR(**16**:*624-639*)
- SKL (L3TLB-GPGPU **640**): L3(**80**:*0-79*), DC(**560**:*80-639*), TX(**560**:*80-639*)

TLB Entry Content

When a page walk entry is cached (or loaded prior to context start), certain bits need to be cached as well along with the physical address bits. The treatment on these bits would be considered when a HIT vs MISS decision needs to be made during a look up.

The purpose of caching is to accelerate the paging process by caching individual translations in **translation look-aside buffers** (**TLBs**). Each entry in a TLB is an individual translation. Each translation is referenced by a page number. It contains the following information from the paging-structure entries used to translate linear addresses with the page number:

- The physical address corresponding to the page number (the page frame).
- The access rights from the paging-structure entries used to translate linear addresses with the page number:
    - The logical-AND of the R/W flags.

    - The logical-AND of the U/S flags.
    - The logical-OR of the XD flags.
- Attributes from a paging-structure entry that identifies the final page frame for the page number (either a PTE or a paging-structure entry in which the PS flag is 1):
    - The dirty flag.
    - The memory type.

PRESENT: This is the same VALID bit description we had in previous page table designs. The lack of present bit (i.e. bit[0]=0) points that rest of the information in the page table entry is being invalid. For some fault models, even NOT PRESENT entries are cached to filter further page faults (*see fault models on caching page faulting entries*). If such entry is cached, there are couple ways that it can be removed from the page tables:

1. LRA selection where the entry becomes a victim for replacement
2. Global or Selective invalidation
3. Page fault response stating the faulting page is now fixed.

R/W Privilege: Certain pages can be allocated as read-only and write operations are not allowed. To make this check work, TLB has to keep the R/W bit. This bit has no effect on read operations; however, for write operation privilege needs to be checked. If there is mis-match, the result of the TLB look-up should be a MISS. This does not mean a page fault immediately; the walk has to be re-done as for any TLB MISS result. There are cases OS may change page table privileges without invalidating pages in TLB (*note: all downgrades result in invalidation of the TLB, however upgrades can be done silently hence re-walk is required*). <u>In case where the TLB Miss is due to privilege mis-match, the existing entry from TLB has to be invalidated and page walk will bring in the most up-to-date copy from memory</u>.

The R/W privilege on final frame is generated as a logical-AND process of all upper page walks pointing to this location.

User vs Supervisor Privilege: The GPU typically operates in user mode when it comes to page tables. So the GTT walk can be treated as faulted when GPU encounters a page with supervisor privileges and the context is marked as user mode. The faulted entry can be cached back into TLB with "P" bit off indicating a faulted entry. However, the page fault report should carry the correct reason why h/w detected the fault in the first place which was the user vs supervisor privilege. There is an option in context header to define the context as supervisor, then it legal to access supervisor pages.

- This is not stored in TLB

The U/S privilege on final frame is generated as a logical-AND process of all upper page walks pointing to this location.

Accessed Bit: This where a stage of the page walk cannot be used if the accessed bit is not set for that level in the page walk. This is true for both storage into TLB as well as to make progress on the page walk. In order to achieve the process of Accessed bit, every stage of the ppGTT read is done via a new semantics between the GAM and GTI such that GTI can atomically process A-bit w/o running into access violations. The details of the semantics are defined as part of the following sections. The “A” bit does not need to be stored as part of the TLB, just the fact that a valid page table entry is present in the TLB does mean that h/w took care of the “A” bit at the time the page was brought up to TLB. Note that TLB prefetching is disabled when A-bit management is enabled.

IA32e mode page tables cannot co-exist with TLB pre-fetching due to lack of A-bit management for all entries of the line.

- This is not stored in TLB

Dirty Bit: Similar to accessed bit, dirty bit needs to be managed. It is only applicable for “write” accesses. Given there are cases where a TLB entry was acquired as part of a read operation, the presence of D-bit should be maintained with the TLB. This gives us the capability to declare a TLB miss for a write access when the D-bit is not set even though TLB has a valid translation. In such case, The TLB entry needs to invalidated and the final stage of the walk needs to be re-done to ensure most up-to-date copy of GTT entry is brought into h/w. The operation of Dirty bit update is also atomic similar to A-bit management.

Execute (XD) Bit: XD bit is also present on every stage of the walk and applicable to executable code that GT would be fetching. In the first pass, instruction cache accesses are not allowed to proceed if the corresponding page does not have the execute credentials set properly. Similar treatment of the TLB entry as privilege bits is expected. A page entry that was already cached in TLB and later accessed for instruction space will have to check the XD bit which is also stored in TLB. If mis-match, the end result is a TLB miss and walk has to be re-done replacing the different stages of the walk.

The XD privilege on final frame is generated as a logical-OR process of all upper page walks pointing to this location.

Faulted Bit: There are usage models where the faulted entries are cached in TLB. This is to filter further faults to the same page as opportunistic way to prevent fault storms. When faulted bit is set the address is included in the TLB look up but final treatment is fault filtering. The rest of the bits are used to define what would be the reason for the fault. If the look-up conflicts with the original faulted reason, a re-walk is required. As a basic case, take a read access bringing up a PTE with W-flag cleared. A subsequent write access has a conflict on privilege, and it will perform a re-walk. If the result of the re-walk is W-flag set, then TLB is upgraded and write makes progress. However, if the result is still W-flag cleared, the write access will fault and TLB entry will be tagged as a faulted entry with only read-aloud. Subsequent write accesses will be filtered as fault but read accesses should cause a re-walk of the page and if successful, the TLB can be updated with PTE as valid with read-only attribute.

TLB Accessed and Dirty Flags

For any paging-structure entry that is used during linear-address translation, bit 5 is the **accessed** flag. For paging-structure entries that map a page (as opposed to referencing another paging structure), bit 6 is the **dirty** flag. These flags are provided for use by memory-management software to manage the transfer of pages and paging structures into and out of physical memory.

Whenever the processor and/or GPU uses a paging-structure entry as part of linear-address translation, it sets the accessed flag in that entry (if it is not already set).

Whenever there is a write to a linear address, the processor and/or GPU sets the dirty flag (if it is not already set) in the paging-structure entry that identifies the final physical address for the linear address (either a PTE or a paging-structure entry in which the PS flag is 1).

Memory-management software may clear these flags when a page or a paging structure is initially loaded into physical memory. These flags are “sticky,” meaning that, once set, the processor and/or GPU does not clear them; only software can clear them.

A processor and/or GPU may cache information from the paging-structure entries in TLBs and paging-structure caches (see Section 4.10). This fact implies that, if software changes an accessed flag or a dirty flag from 1 to 0, the GPU might not set the corresponding bit in memory on a subsequent access using an affected linear address

Accessed bit is applicable to every stage of the page walk, however the dirty bit is only applicable to final stage of the walk.

The rule states that a particular access cannot be committed until the Accessed and/or Dirty bits are not visible to page management s/w. In order for GPU to follow the rule, GTT accesses (when A/D bits are supported) are going to be done via a special cycle definition between GAM and GTI.

Updating A/D Bits

New atomic operations are added to GAM to GPU interface (GTI) to handle paging entries. GAM has to set the correct atomic opcodes based on the access type and context entry controls as well as level of access.

Requires setting for opcodes are given in the table below. The steps of operations in the atomic ALUs are given later in the document.

The Following Atomics are only applicable in GTI and used for Page Walks

R/W => Bit[0]

Extended Access required => Bit[1]

Write Protect Enable => Bit[2]

Intermediate Entry => Bit[3]

Atomic Operation	Opcode	Description	New Destination Value	Applicable	Return Value (optional)
Atomic_Page_update_0000	1100_0000	Read Access Extended Access bit is disabled Write Protection is disabled Final PTE	Set bit[5] if not set		new_dst
Atomic_Page_update_0001	1100_0001	Write Access Extended Access bit is disabled Write Protection is disabled Final PTE	Set bit[5,6] if not set		new_dst
Atomic_Page_update_0000	1100_0010	Read Access Extended Access bit is enabled Write Protection is disabled Final PTE	Set bit[5,10] if not set		new_dst
Atomic_Page_update_0001	1100_0011	Write Access Extended Access bit is enabled	Set bit[5,6,10] if not set		new_dst

		Write Protection is disabled Final PTE			
Atomic_Page_update_0100	1100_0100	Read Access Extended Access bit is disabled Write Protection is enabled Final PTE	Set bit[5] if not set		new_dst
Atomic_Page_update_0101	1100_0101	Write Access Extended Access bit is disabled Write Protection is enabled Final PTE	Set bit[5,6] if not set		new_dst
Atomic_Page_update_0100	1100_0110	Read Access Extended Access bit is enabled Write Protection is enabled Final PTE	Set bit[5,10] if not set		new_dst
Atomic_Page_update_0101	1100_0111	Write Access Extended Access bit is enabled Write Protection is enabled Final PTE	Set bit[5,6,10] if not set		new_dst
Atomic_Page_update_0000	1100_1000	Read Access Extended Access bit is disabled Write Protection is disabled Intermediate Paging Entry	Set bit[5] if not set		new_dst

Atomic_Page_update_0001	1100_1001	Write Access Extended Access bit is disabled Write Protection is disabled Intermediate Paging Entry	Set bit[5,6] if not set		new_dst
Atomic_Page_update_0000	1100_1010	Read Access Extended Access bit is enabled Write Protection is disabled Intermediate Paging Entry	Set bit[5,10] if not set		new_dst
Atomic_Page_update_0001	1100_1011	Write Access Extended Access bit is enabled Write Protection is disabled Intermediate Paging Entry	Set bit[5,6,10] if not set		new_dst
Atomic_Page_update_0100	1100_1100	Read Access Extended Access bit is disabled Write Protection is enabled Intermediate Paging Entry	Set bit[5] if not set		new_dst
Atomic_Page_update_0101	1100_1101	Write Access Extended Access bit is disabled Write Protection is enabled Intermediate Paging Entry	Set bit[5,6] if not set		new_dst

Atomic_Page_update_0100	1100_1110	Read Access Extended Access bit is enabled Write Protection is enabled Intermediate Paging Entry	Set bit[5,10] if not set		new_dst
Atomic_Page_update_0101	1100_1111	Write Access Extended Access bit is enabled Write Protection is enabled Intermediate Paging Entry	Set bit[5,6,10] if not set		new_dst

Atomic updates are only possible for cacheable memory types. There could be cases where the PTE could be in WT/WC/UC space where atomic update is not possible via WB space. Those are the cases where IA cores use bus lock to update the A/D bits in PTE.

GT core is not capable of supporting bus locks and has the following options. These options will be enabled/disabled via register space.

Option#1: Ignore the PAT/MTRR setting of the PTE and update the space as WB with atomic ops. This is the place GAM will decide to go forward with atomic updates assuming WB space works

Option#2: Once the memory type is determined and the end result of the page is WC/UC/WT space, we cannot guarantee an atomic update. GAM will report an application error (catastrophic) to the scheduler and handle the case as error.

Bit	Access	Default Value	Description
1	R/W	0b	**A/D Bit Update on non-WB Space:** A/D bit updates are only possible via atomic operations which are required to be on WB space to work properly. On non-WB spaces, the A/D bit updates are done via bus locks which are not supported for GT. **"1"**: Ignore the page level cacheability and do atomic updates for A/D bit management **"0"**: Detect the page level cacheability as part of the atomic operation and throw a catastrophic error when non-WB space is seen for A/D bit updates.

Replacement

TLB replacements during runtime are based on LRA algorithm; in addition, invalidations and page responses will have to invalidate the TLB entries.

Invalidations of TLB

There are various ways to invalidate TLBs:

1. **Traditional invalidation from command streamer**: Could be part of any fence accesses including newly added atomics
2. **SVM based invalidations**: Listed as part of the new SVM related invalidations, various stages of TLBs including intermediate stages can be invalidated selectively and/or as a whole.
3. **Context Switch**: A context switch has to invalidate caches to make sure we have no residual value of the TLBs across multiple PASIDs. GAM will treat the context reload message from CS as a form of TLB invalidation.
4. **A page response**: should invalidate faulted recordings. It should be done via address matching to kick the faulted entries within the matching PASID.

Invalidation response "Invalidation Wait Descriptor" should also be a fence for both READs and WRITEs that used the previous TLB entries. GAM can only respond to "invalidation wait descriptor" after getting a GTI EMPTY indication.

Optional Invalidations

The following cases are listed as page table updates which software may choose not to invalidate the TLBs.

- If a paging-structure is modified to change the Present (Valid) flag from 0 to 1, s/w may choose not to invalidate TLBs. This affects only the case where GPU keeps the faulted page in its TLB to filter out future faults. Regardless of s/w does invalidation or not, for the cases where h/w cares, there will be a page response from s/w which will be used to shootdown the faulted record from the TLB.

  GAM will put faulted entries to its TLBs only if there has been page request for it, which means that only faultable surfaces can be stored in GAM TLBs as a faulted entry.

- If a paging-structure entry is modified to change the accessed flag from 0 to 1,no invalidation is necessary (assuming that an invalidation was performed the last time the accessed flag was changed from 1 to 0). This is because no TLB entry or paging-structure cache entry is created with information from a paging structure entry in which the accessed flag is 0.
- If a paging-structure entry is modified to change the R/W or U/S or XD flag from 0 to 1, failure to perform an invalidation may result in a "spurious" page-fault exception (e.g., in response to an attempted write access) but no other adverse behavior. Such an exception will occur at most once for each affected linear address

Atomic Operations between GPU and IA

IA cores are capable of doing atomic operations on any memory space defined as part of their page tables, MTRRs, and PAT. The core detects the memory type after applying all the checks, and if the end target is within WB space, it uses the optimized MESI protocol (i.e. RFO, WB) to complete the atomic operation. If the memory space is UC/WC/WT, it uses the bus lock to honor the atomic requirements.

GPUs are only capable of doing atomics via optimized MESI protocols; they do not have a mechanism to do bus locking. This restricts atomic operations to WB space if they happened to be between GPU and IA cores. We cannot guarantee the atomicity for UC/WC/WT.

There are multiple ways how GPU can handle atomics to non-WB space.

1. **Ignore the Memory Type:** GPU ignores the memory type for the atomic operation and performs it via RFO/WBMtol as if the space is WB.

   If IA is relying on the LOCK for the same atomic, LOCK keeps the GPU off the bus for integrated solutions. And if GPU has the ownership of the line, a LOCKed read from IA still evicts the line from GT even if the IA has the line as UC/WT/WC in its memory space.
2. **Report as an error and fail the cycle:** GPU detects the mismatch and handles the cycle as error (i.e. write has no affect and read returns garbage) and reports the error as catastrophic to SW. App can be terminated by SW.
3. **Perform Bus Lock:** Similar to IA cores, GPU can support bus locks around atomic operations to UC/WT/WC spaces. However, the initial implementations of shared atomic implementations do not have this option in GPU hardware.

Option#2 is the main mode where GPU is not expecting atomics to UC/WT/WC space and any such accesses are faulted as HW cannot guarantee the atomicity of the operation with the current implementation.