

# **Intel® Iris® Xe MAX Graphics Open Source**

## **Programmer's Reference Manual**

**For the 2020 Discrete GPU formerly named "DG1"**

Volume 6: Memory Views

February 2021, Revision 1.0



## Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks.

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

## Table of Contents

<b>Memory Views.....</b>	<b>1</b>
Introduction.....	1
Memory Views Glossary.....	3
Graphics Virtual Memory.....	3
Graphics Translation Tables.....	3
<b>Virtual Addressed TR Translation Tables .....</b>	<b>10</b>
Walk with 64KB Page.....	18
PDE for Page Table.....	19
PTE: Page Table Entry for 64KB Page.....	20
PTE: Page Table Entry for 4KB Page.....	21
Walk with 64KB Page.....	23
Walk with 2MB Page.....	24
Pointer to PML4 table.....	26
PML4E: Pointer to PDP Table.....	26
PDPE: Pointer to PD Table.....	26
PDPE for PD.....	27
PDPE for 1GB Page.....	28
PD: Pointer to Page Table.....	29
PDE for Page Table.....	29
PDE for 2MB Page.....	30
PTE: Page Table Entry for 64KB Page.....	31
PTE: Page Table Entry for 4KB Page.....	32



## Memory Views

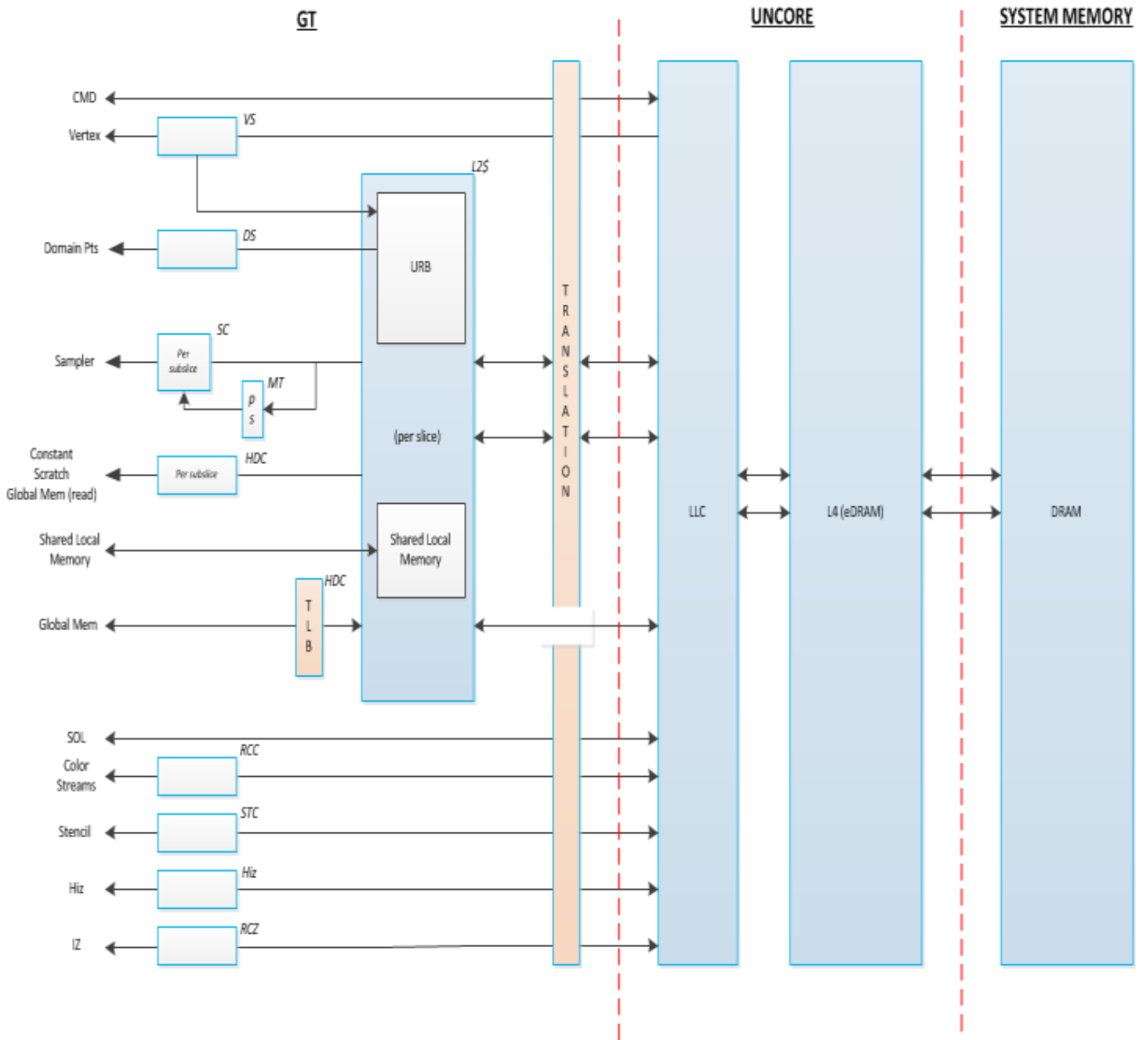
### Introduction

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.

## Cache and Memory Hierarchy



## 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 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 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 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

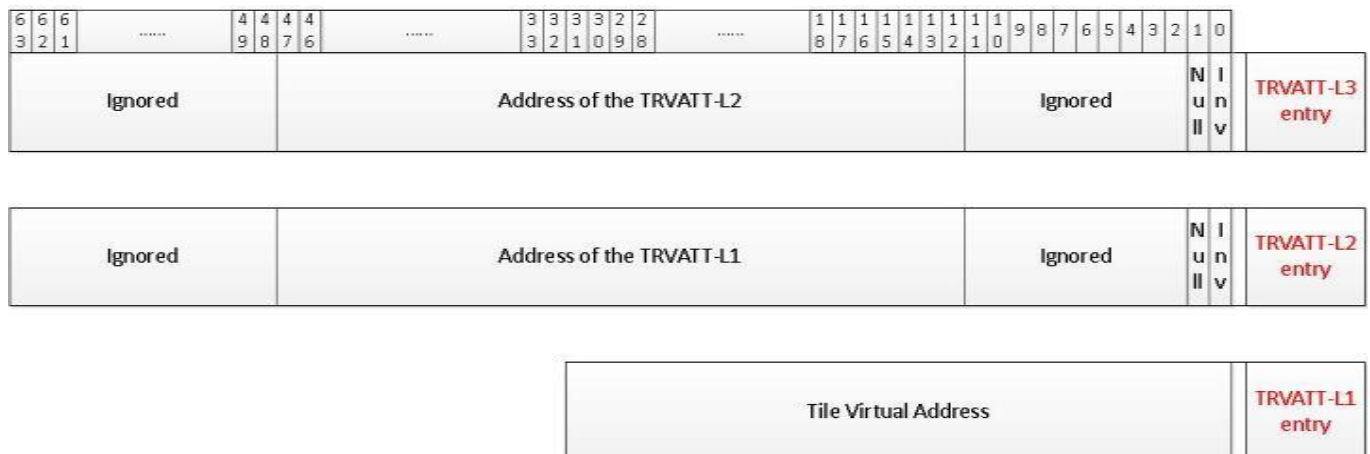
All page tables use the same 64-bit PTE format. Differences are in how various bit fields applies (vs reserved) under various usage models.

Follows the same principles that were 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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L1. <i>TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.</i>
11:2	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L2. <i>TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.</i>
11:2	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	
<b>Context:</b>	Tiled ResourceTranslation Tables in Gfx Page Tables
Usage model for TR translations are restricted to GFX Render Engine (& POSH pipeline).	

Programming Note	
<b>Context:</b>	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.	

Programming Note	
<b>Context:</b>	Tiled ResourceTranslation Tables in Gfx Page Tables
When partitioned address space based Dual Context is enabled, bit[47] of the virtual address in L3, L2 and L1 entries must be 0.	

### 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						
<b>Register Space:</b>		MMIO: 0/2/0				
DWord	Bit	Description				
1	63:48	<p><b>Reserved</b></p> <table border="1"> <tr> <td><b>Access:</b></td> <td>RO</td> </tr> </table> <p>Reserved.</p>	<b>Access:</b>	RO		
	<b>Access:</b>	RO				
47:32	<p><b>Tiled Resource - VA translation Table L3 Pointer (Upper Address)</b></p> <table border="1"> <tr> <td><b>Default Value:</b></td> <td>0000h</td> </tr> <tr> <td><b>Access:</b></td> <td>R/W</td> </tr> </table> <p>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.</p>	<b>Default Value:</b>	0000h	<b>Access:</b>	R/W	
<b>Default Value:</b>	0000h					
<b>Access:</b>	R/W					
0	31:16	<p><b>Tiled Resource - VA translation Table L3 Pointer (Lower Address)</b></p> <table border="1"> <tr> <td><b>Default Value:</b></td> <td>0000h</td> </tr> <tr> <td><b>Access:</b></td> <td>R/W</td> </tr> </table> <p>Lower address bits for tiled resource VA to virtual address translation L3 table.</p>	<b>Default Value:</b>	0000h	<b>Access:</b>	R/W
		<b>Default Value:</b>	0000h			
<b>Access:</b>	R/W					

Tiled Resources VA Translation Table L3 Pointer				
	15:0	<p><b>Reserved</b></p> <table border="1"> <tr> <td><b>Access:</b></td> <td>RO</td> </tr> </table> <p>Reserved.</p>	<b>Access:</b>	RO
<b>Access:</b>	RO			

Tiled Resources Null Tile Detection Register						
<b>Register Space:</b>		MMIO: 0/2/0				
DWord	Bit	Description				
	31:0	<p><b>Null Tile Detection Value</b></p> <table border="1"> <tr> <td><b>Default Value:</b></td> <td>00000000h</td> </tr> <tr> <td><b>Access:</b></td> <td>R/W</td> </tr> </table> <p>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.</p>	<b>Default Value:</b>	00000000h	<b>Access:</b>	R/W
<b>Default Value:</b>	00000000h					
<b>Access:</b>	R/W					

Tiled Resources Invalid Tile Detection Register						
<b>Register Space:</b>		MMIO: 0/2/0				
DWord	Bit	Description				
	31:0	<p><b>Invalid Tile Detection Value</b></p> <table border="1"> <tr> <td><b>Default Value:</b></td> <td>00000000h</td> </tr> <tr> <td><b>Access:</b></td> <td>R/W</td> </tr> </table> <p>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.</p>	<b>Default Value:</b>	00000000h	<b>Access:</b>	R/W
<b>Default Value:</b>	00000000h					
<b>Access:</b>	R/W					

Tiled Resources Virtual Address Detection Registers (TRVADR)					
<b>Register Space:</b>		MMIO: 0/2/0			
DWord	Bit	Description			
0	31:8	<p><b>Reserved</b></p> <table border="1"> <tr> <td><b>Access:</b></td> <td>RO</td> </tr> </table> <p>Reserved.</p>	<b>Access:</b>	RO	
	<b>Access:</b>	RO			
7:4	<p><b>TRVA Mask Value (TRVAMV)</b></p> <table border="1"> <tr> <td><b>Default Value:</b></td> <td>0000b</td> </tr> <tr> <td><b>Access:</b></td> <td>R/W</td> </tr> </table>	<b>Default Value:</b>	0000b	<b>Access:</b>	R/W
<b>Default Value:</b>	0000b				
<b>Access:</b>	R/W				

Tiled Resources Virtual Address Detection Registers (TRVADR)					
	<p>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 (<i>this field defaults to "0000" to disable detection</i>)</p> <p><i>Note that h/w supports two possible values for MASK: "0000" which is disabled case and "1111" where 44 bit TR-VA space is carved out.</i></p>				
3:0	<p><b>TRVA Data Value (TRVADV)</b></p> <table border="1"> <tr> <td><b>Default Value:</b></td> <td>0b</td> </tr> <tr> <td><b>Access:</b></td> <td>R/W</td> </tr> </table> <p>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.</p>	<b>Default Value:</b>	0b	<b>Access:</b>	R/W
<b>Default Value:</b>	0b				
<b>Access:</b>	R/W				

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

The following register (0x4DFC[0]) has enable and disable control of the bypass path across TR translations. 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	
<b>Context:</b>	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	
<b>Context:</b>	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.

*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**

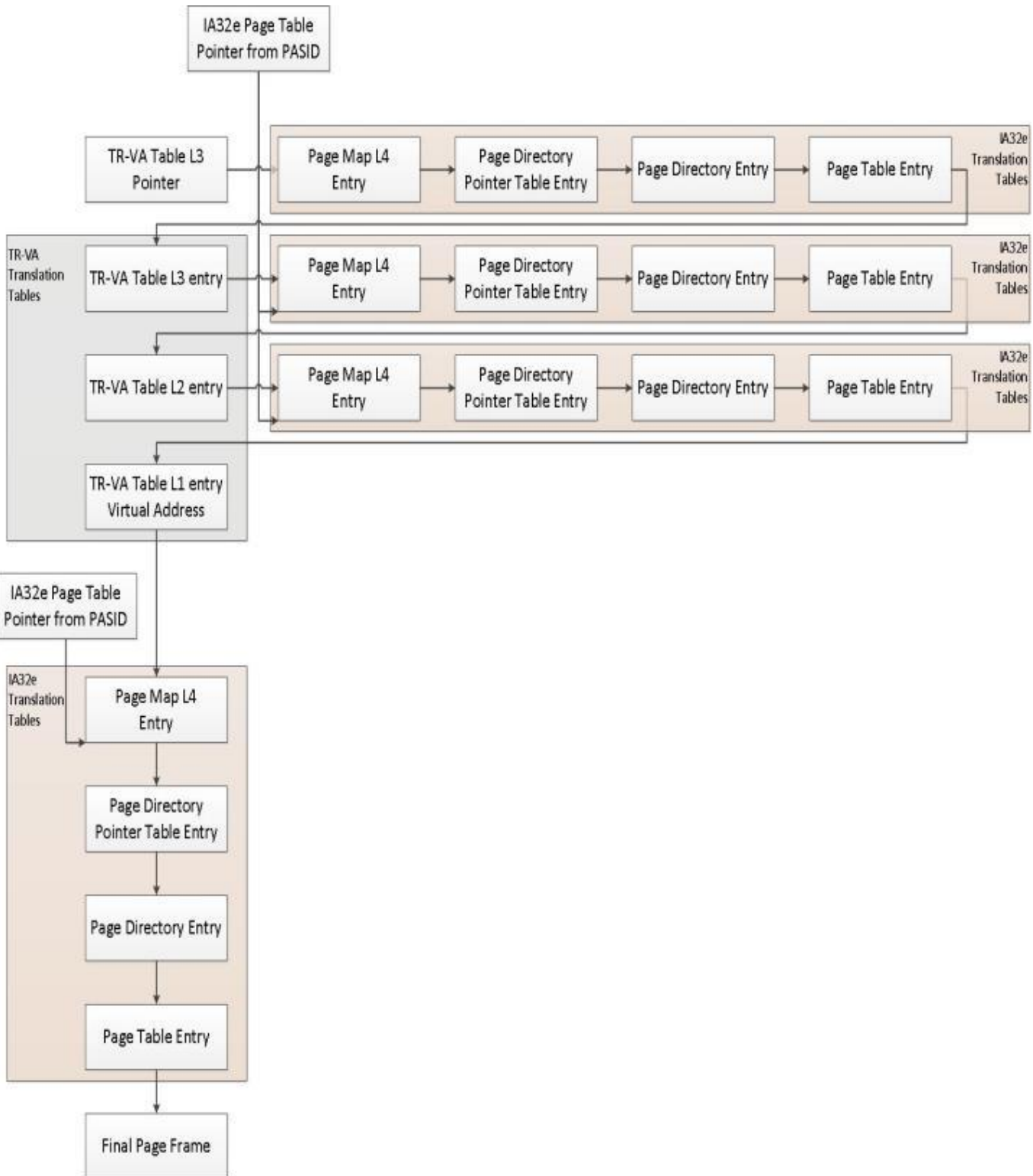
The L3 table pointer along with TRTTL3e/TRTTL2e is projected to support two modes of address space. Original intent was to have the contents to be in Virtual Address space (OS managed) and have them to be translated to GPA to HPA before getting accessed. Such mechanism will incur high latency penalties 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 1<sup>st</sup> 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 2<sup>nd</sup> Level (VTd) translations enabled.



Once 2<sup>nd</sup> level translations are enabled each level of 1<sup>st</sup> 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.



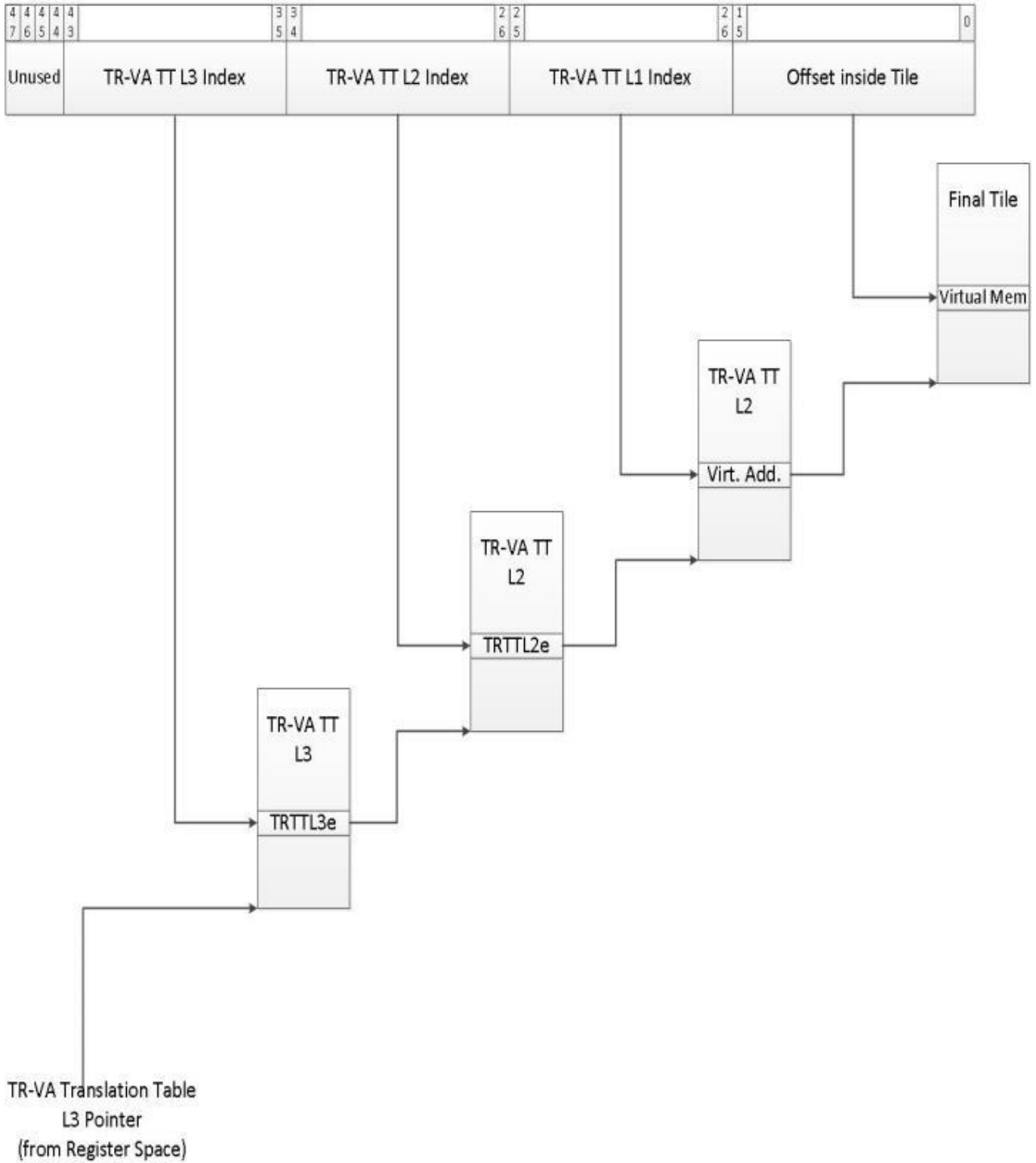
## TR-TT Page Walk

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.





## 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 software (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 software 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.

## Global GTT

The Global GTT mechanism looks very similar to previous 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 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:

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 to Final Page (4KB)														Ignored				P	Global GTT Entry																	

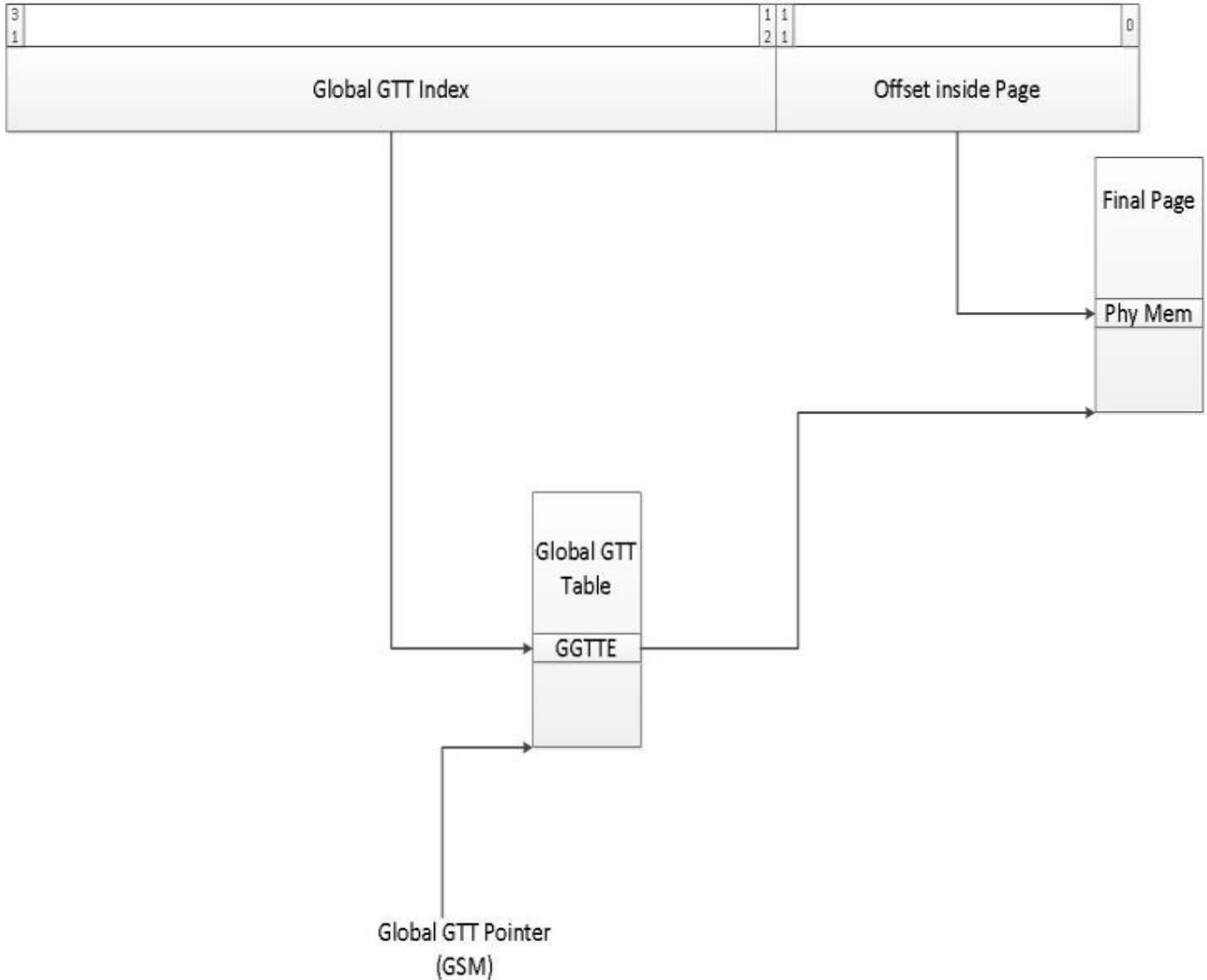
Bits	Field	Description
63:57	Ignored	Ignored (h/w does not care about values behind ignored registers)
51:HAW	Ignored	Ignored (h/w does not care about values behind ignored registers)
(HAW-1):12	Address	Physical address of 4KB memory page referenced by this entry.
11:5	Ignored	Ignored (h/w does not care about values behind ignored registers)
4:2	Function Number	The Number of the Function to which this page has been assigned. This field is ignored unless HW Graphics Virtualization (SRIOV) is enabled.
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. 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.



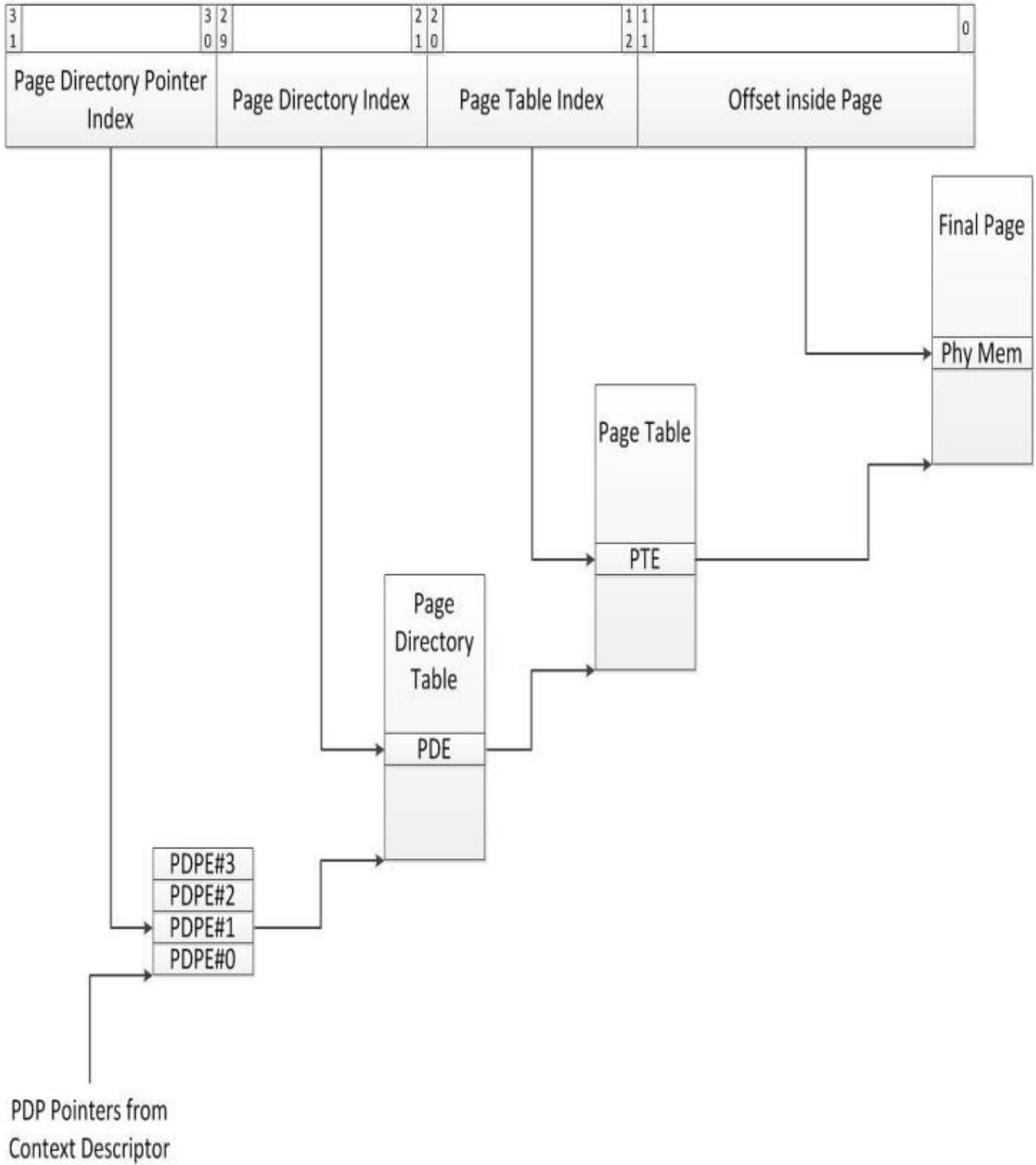
### Per-Process GTT with 32b VA

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

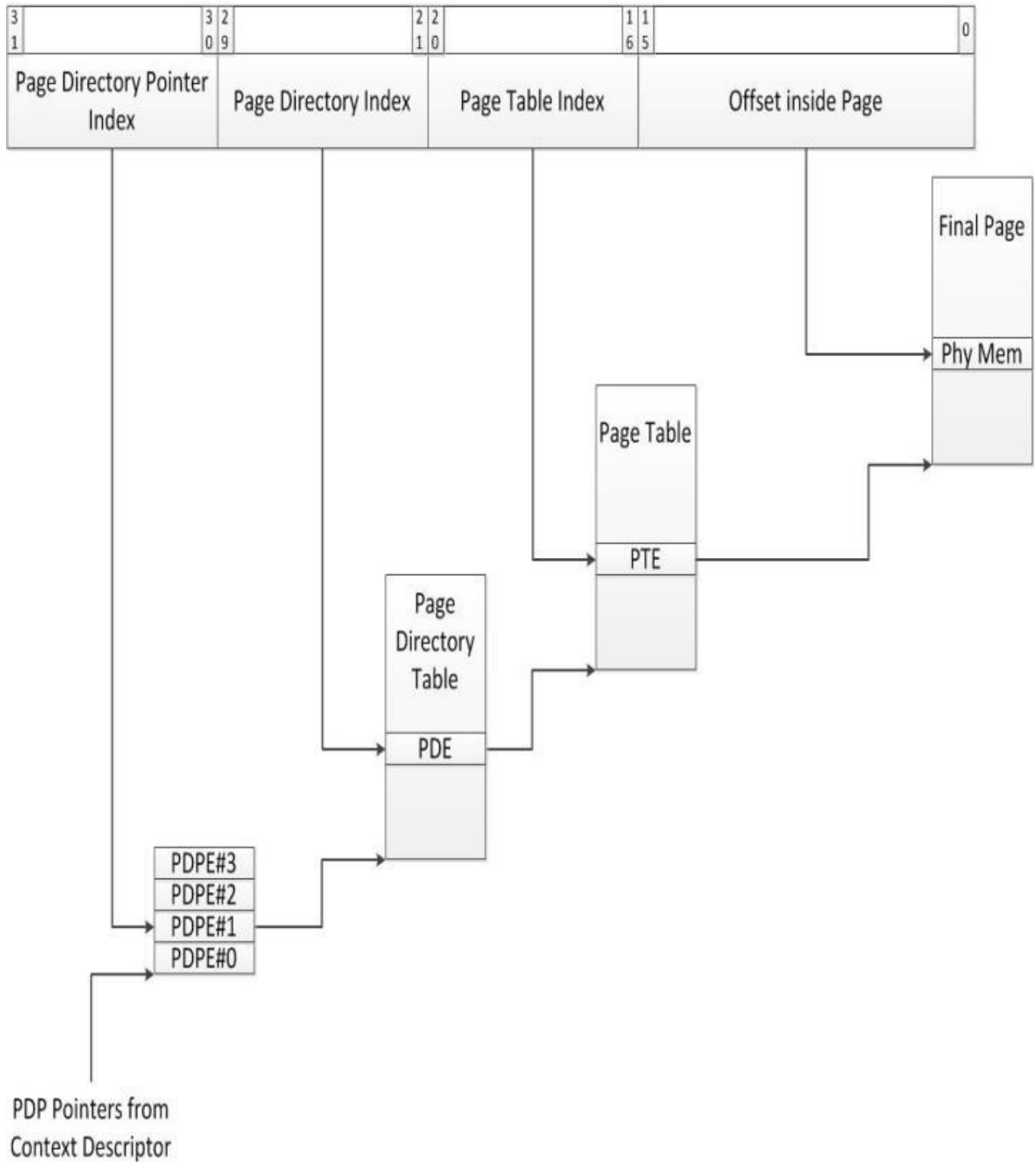
### Page Walk in Legacy mode with 32b VA

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.



## Walk with 64KB Page



## Page Table Entry (PTE) Formats

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

6 6 6 6 5 5 5 5 5 5 5 5 3 2 1 0 9 8 7 6 5 4 3 2 1	HAW	HAW-1	3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0	Ignored	Address of Page Table	Ignored	R / P W	PDE
Ignored	Address of the 64KB MB page			Rsvd.	N I P I g n u g A l l n T	I g n	P P I R C W g / P D T n W	PTE (64KB Page)
Ignored	Address of 4KB Page			I g n	N I P u g A l l n T	I g n	P P I R C W g / P D T n W	PTE (4KB Page)

## PDE for Page Table

6 6 6 6 5 5 5 5 5 5 5 5 3 2 1 0 9 8 7 6 5 4 3 2 1	HAW	HAW-1	3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0	Ignored	Address of page-directory-pointer-table	Ignored	R / P W	PDE
--	-----	-------	--	---------	---	---------	------------	-----

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. <b>In Legacy mode with 32b VA, R/W bits from PDE are not used.</b>
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

63:10	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 the 64KB MB page													Rsvd.	Ign	N	I	P	Ign	P	P	I	R	P	PTE (64KB Page)							

Bits	Field	Description
63:HAW*	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
(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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1 <sup>st</sup> 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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege ( <i>and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry</i> ) to the memory region controlled by this entry. See a later section for access rights. <i>GPU does not support Supervisor mode contexts.</i>
0	P: Present	It must be "1" to point to a 64KB Page.

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

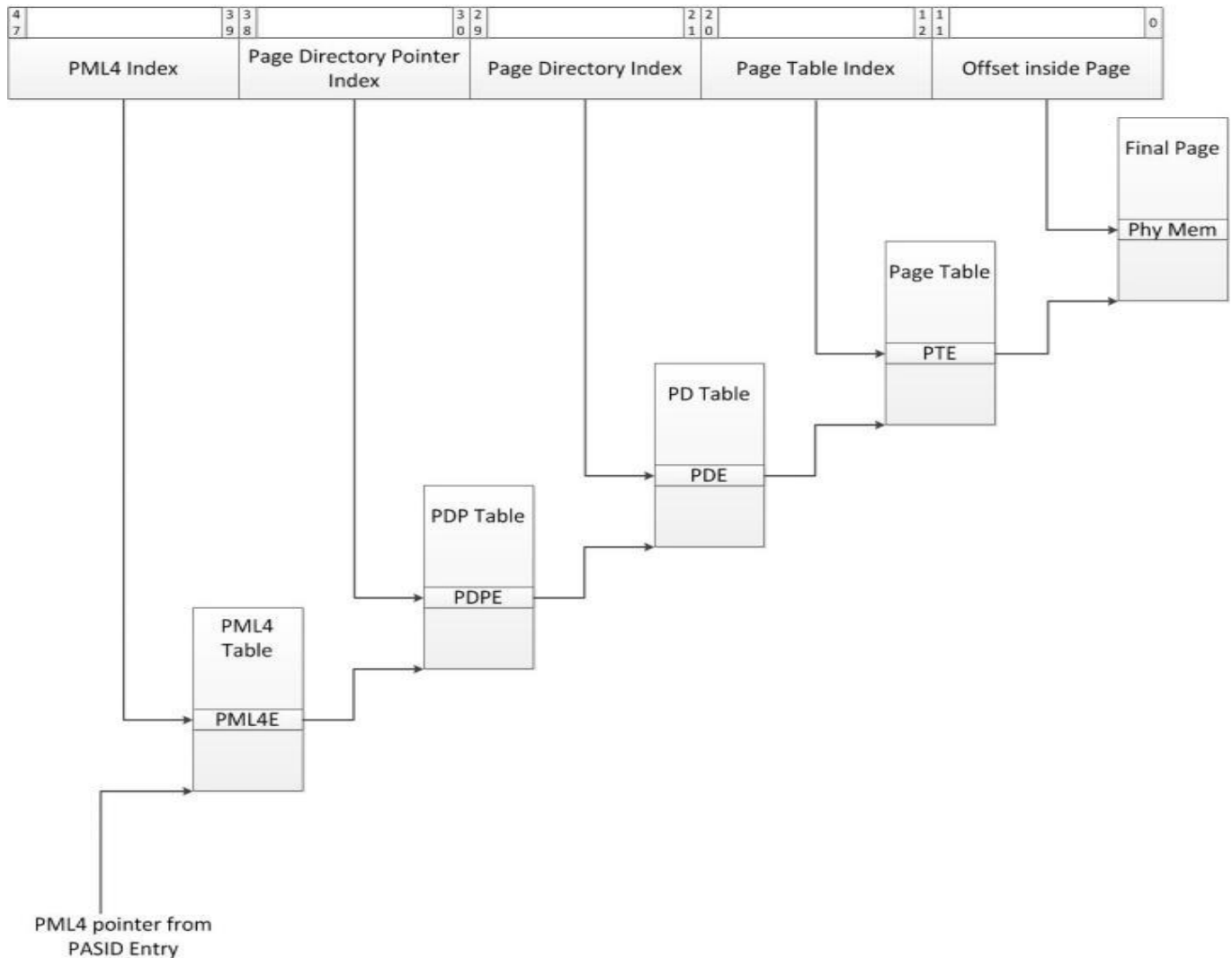




### Page Walk in Legacy 48b Mode

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.

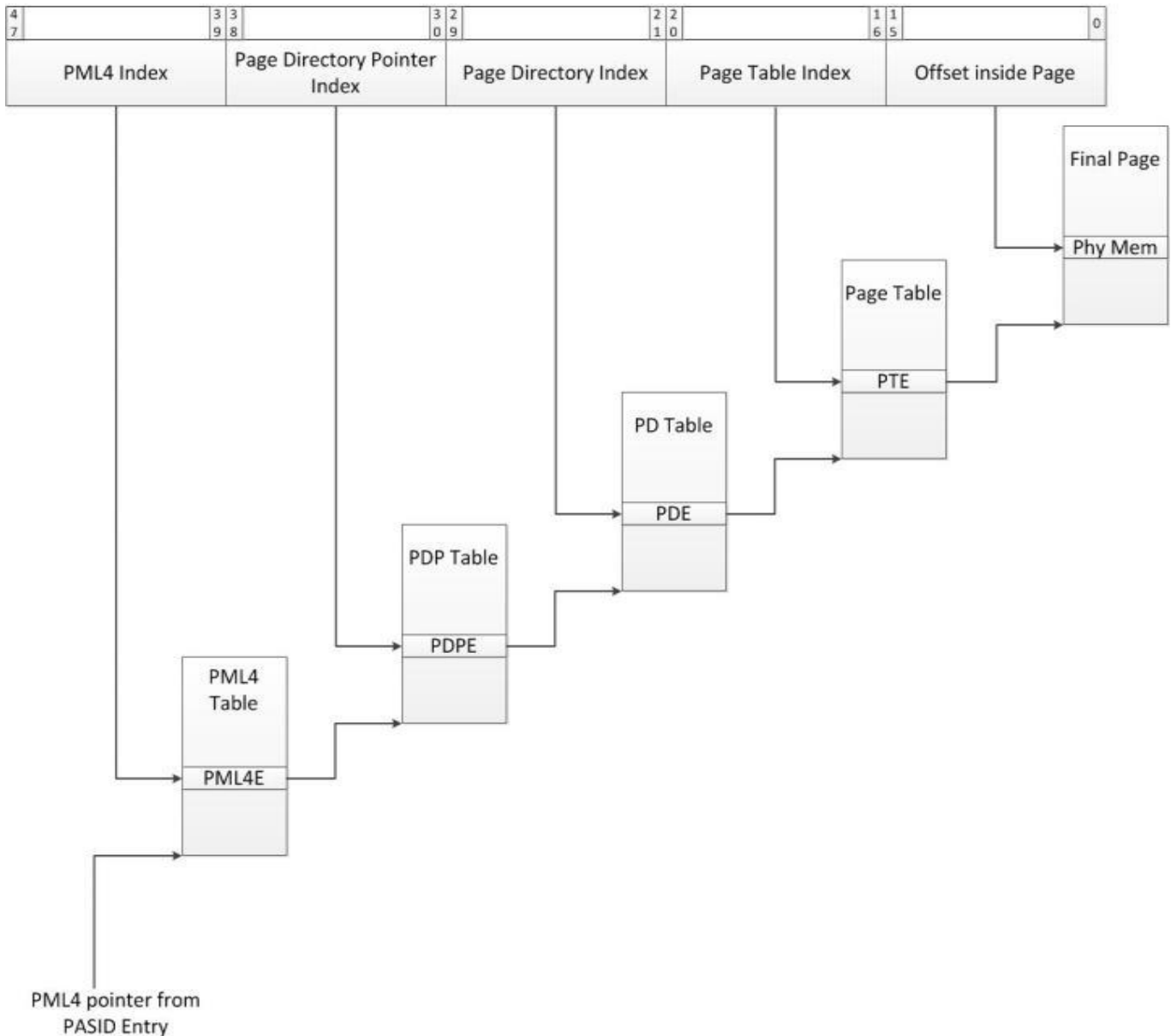
To repurpose the caches the following mechanism will be used:

- 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.

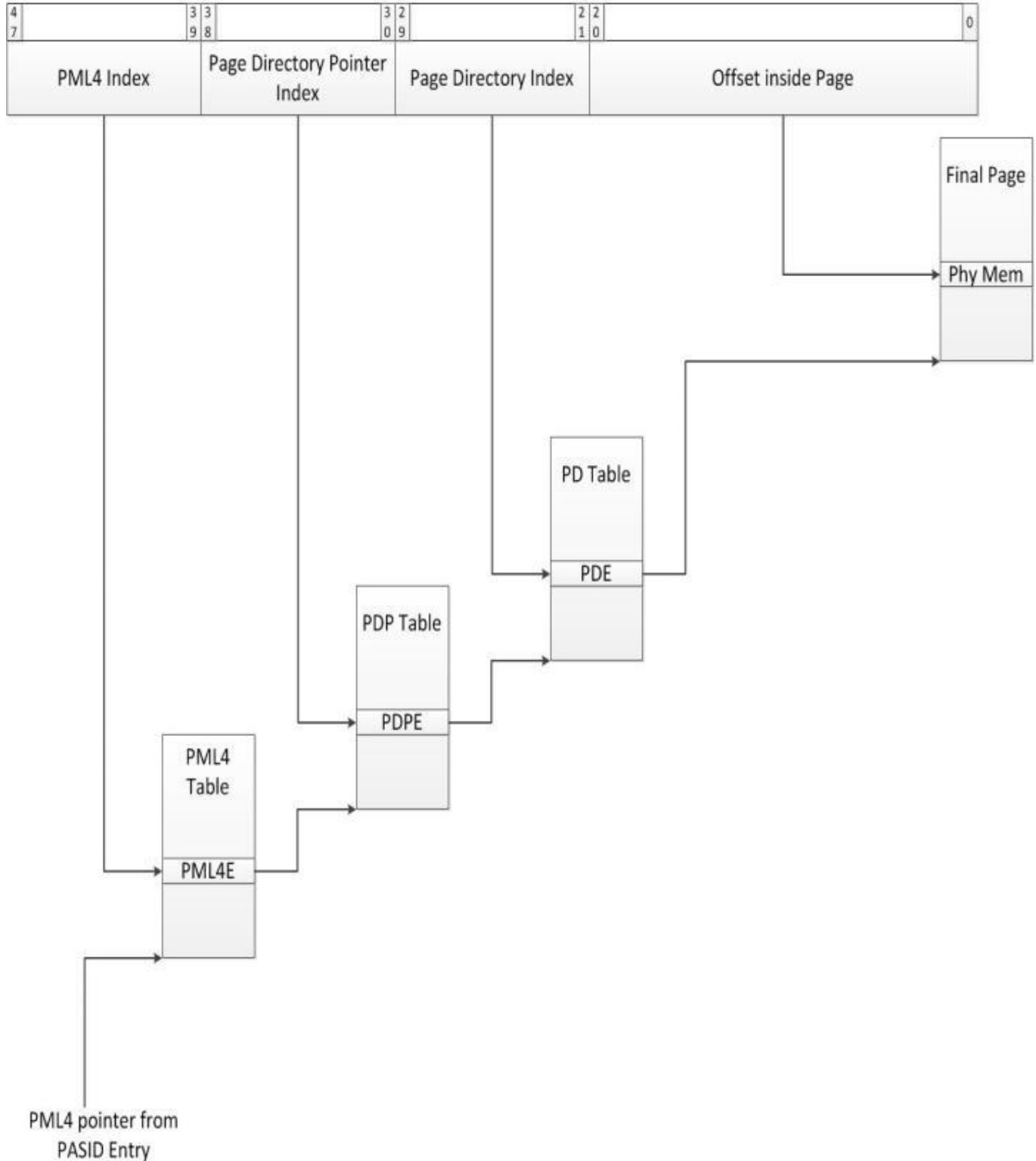
## Walk with 64KB Page

64KB Page size has a slightly different usage for how PTEs are selected for the corresponding 64KB page. In page table every 16<sup>th</sup> 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 16<sup>th</sup> 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

PPGTT32 does not support 1GB pages.

### Page Tables Entry PTE Formats

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

6	6	6	6	5	5	5	5	5	5	5	5			HAW	HAW-1		3	3	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0																													
Ignored																																	Address of page-directory-pointer-table														Ignored														R	/	P	W	PML4E															
Ignored																																	Address of the 1GB page														Reserved														P	A	Ign	N	I	P	1	Ign	P	P	I	R	P	D	W	g	/	P	W	PDPE (1GB Page)
Ignored																																	Address of page-directory-table														Ignored														R	/	P	W	PDPE Page Directory															
Ignored																																	Address of the 2MB page														Reserved														P	A	Ign	N	I	P	1	Ign	P	P	I	R	P	D	W	g	/	P	W	PDE (2MB Page)
Ignored																																	Address of page-table														Ignored														R	/	P	W	PDE Page Table															
Ignored																																	Address of the 64KB MB page														Rsvd.														Ign	N	I	P	1	Ign	P	P	I	R	P	D	W	g	/	P	W	PTE (64KB Page)		
Ignored																																	Address of 4KB Page														Ign														N	I	P	1	Ign	P	P	I	R	P	D	W	g	/	P	W	PTE (4KB Page)			

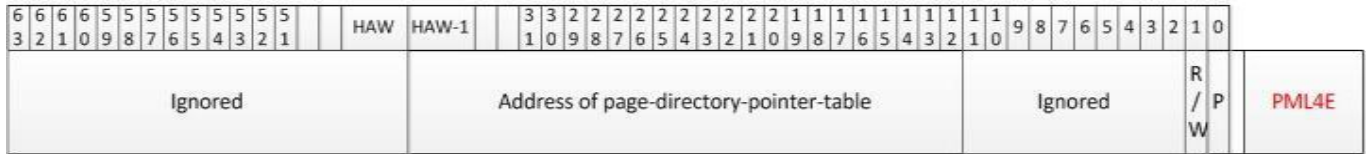
Each table entry is further broken down along with the required functions. GFX has a 4-level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required 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. <i>GPU does not support Supervisor mode contexts.</i>
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

6	6	6	5	5	5	5	5	5	5	5			HAW	HAW-1			3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
Ignored													Address of page-directory-table													Ignored										R	P	PDPE Page Directory									
																																				W											

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.





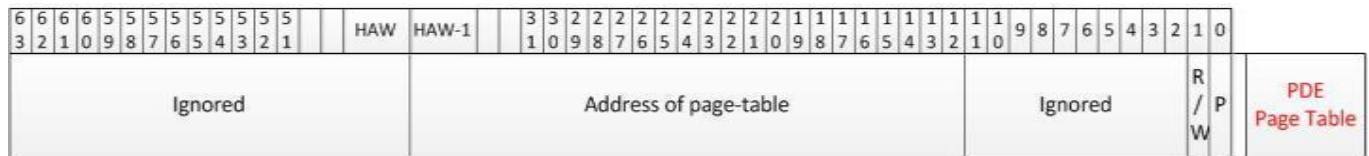
\* HAW = 39 for client, and 46 for server.

## 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. <i>GPU does not support Supervisor mode contexts.</i>
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

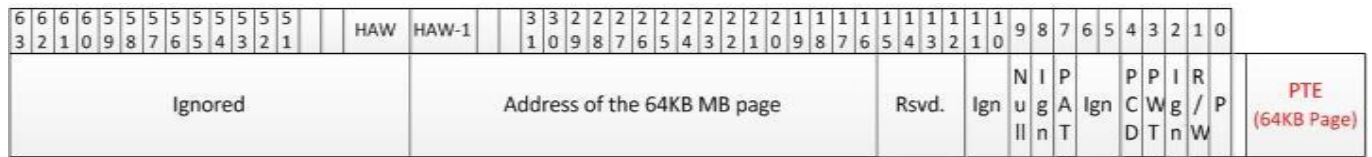
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Ignored												Address of the 2MB page												Reserved												PAT	N	I	g	1	Ign	P	C	W	g	/	P	I	R	P	D	T	n	W	PDE (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 (1 <sup>st</sup> 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

Bits	Field	Description
		<i>extended-context-entry</i> ) to the memory region controlled by this entry. See a later section for access rights.  <i>GPU does not support Supervisor mode contexts.</i>
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*	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
(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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1 <sup>st</sup> 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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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

Bits	Field	Description
		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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege ( <i>and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry</i> ) to the memory region controlled by this entry. See a later section for access rights.  <i>GPU does not support Supervisor mode contexts.</i>
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*	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
(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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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	<i>Ignored</i>	<i>Ignored (h/w does not care about values behind ignored registers)</i>
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.

## 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.