# Intel® UHD Graphics Open Source

# Programmer's Reference Manual

## For the 2020 Intel Core™ Processors with Intel Hybrid Technology based on the "Lakefield" Platform

Volume 6: Memory Views

April 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.

# Table of Contents

# 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 | |
|---|---|---|---|
| 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.<br><br>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. Software may read PTE values from the global Graphics Translation Table GTT. PTEs cannot be written directly into | This is a 16MB bar above physical memory. | 16 MB<br><br>(2 MB MMIO + 6 MB reserved + 8 MB GGTT) |

| Address Type | Description | Range | |
|---|---|---|---|
| | the global GTT memory area.<br>GTTMMADR must be marked uncache-able (UC).<br>Accesses to GTTMMADR(GTT) must be naturally aligned and 64 bits or less (ie. 1 GTT entry).<br>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.<br>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.<br>This is a stolen memory which can be accessed via GMADR for CPU and directly for GPU/DE.<br>Size is programmable with 32 MB multiplier.<br>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. | |

# 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 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

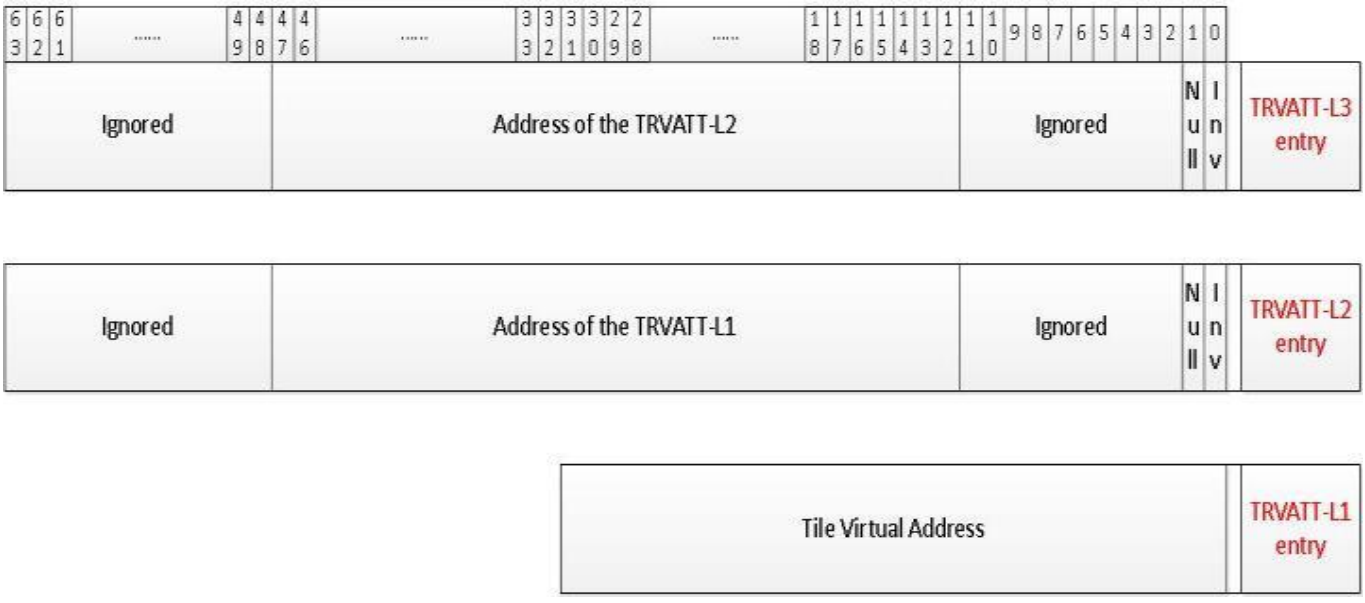
- IA32e compatible GTT
- PPGTT – private per process GTT (private GFX)
- GGTT - global GTT

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

## 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.<br>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.<br>*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.<br>*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**<br>**Access:** RO<br>Reserved. |
| | 47:32 | **Tiled Resource – VA translation Table L3 Pointer (Upper Address)**<br>**Default Value:** 0000h<br>**Access:** R/W<br>Upper address bits for tiled resource VA to virtual address translation L3 table.<br>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)**<br>**Default Value:** 0000h<br>**Access:** R/W<br>Lower address bits for tiled resource VA to virtual address translation L3 table. |
| | 15:0 | **Reserved**<br>**Access:** RO<br>Reserved. |

| Tiled Resources Null Tile Detection Register | | |
|---|---|---|
| **Register Space:** | | MMIO: 0/2/0 |
| **DWord** | **Bit** | **Description** |
| | 31:0 | **Null Tile Detection Value**<br>**Default Value:** 00000000h<br>**Access:** R/W<br>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**<br>**Default Value:** 00000000h<br>**Access:** R/W<br>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**<br>**Access:** RO<br>Reserved. |
| | 7:4 | **TRVA Mask Value (TRVAMV)**<br>**Default Value:** 0000b<br>**Access:** R/W<br>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*)<br>*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.* |

| Tiled Resources Virtual Address Detection Registers (TRVADR) | | |
|---|---|---|
| | 3:0 | **TRVA Data Value (TRVADV)**<br>**Default Value:** 0b<br>**Access:** R/W<br>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**<br>**Access:** RO<br>Reserved. |
| | 1 | **TR-VA Translation Table Memory Location**<br>**Default Value:** 0b<br>**Access:** R/W<br>This fields specifies whether the translation tables for TR-VA to VA are in virtual address space vs physical (GPA) address space.<br>0: Tables are in Physical (GPA) Space<br>1: Tables are in Virtual Address Space<br>**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).** |
| | 0 | **TR-TT Enable**<br>**Default Value:** 0b<br>**Access:** R/W<br>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. 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 invallide 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 Invalida 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 convertes 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

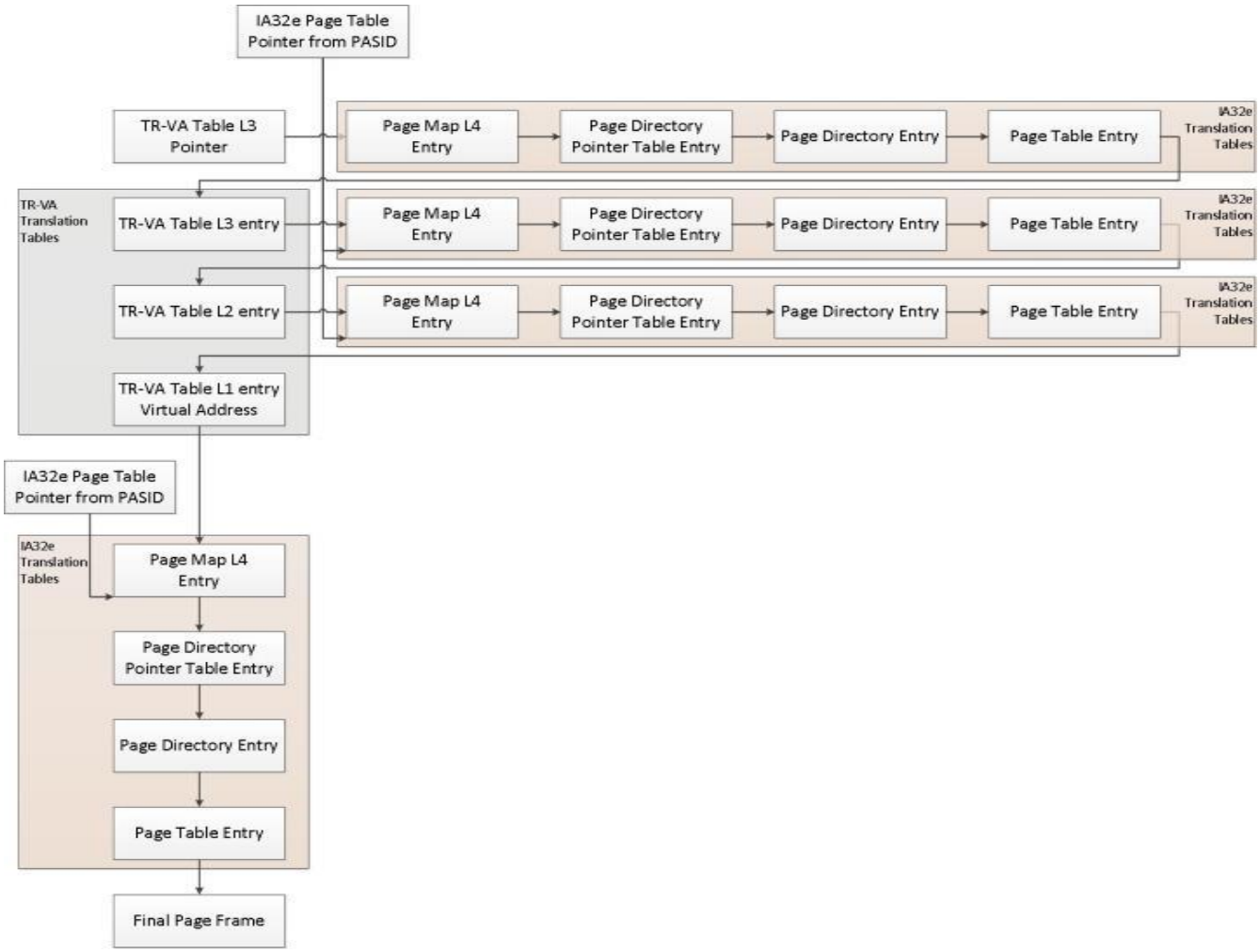
The L3 table pointer along with TRTTL3e/TRTTL2e is projected to support two modes of address space. Original intent was to have the contents to be in Virtual Address space (OS managed) and have them to be translated to GPA to HPA before getting accessed. Such mechanism will incur high latency penalties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

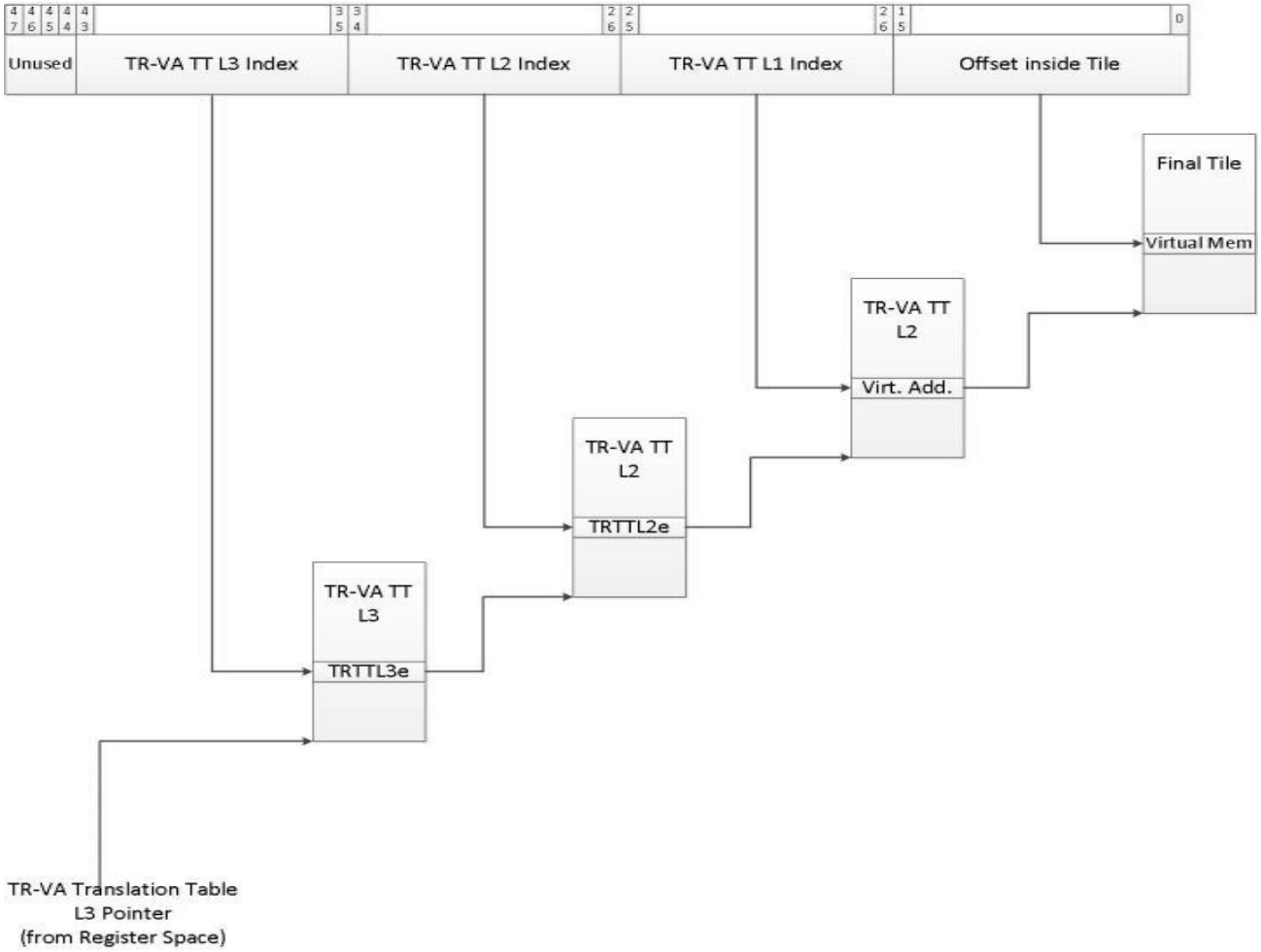
The level of nested walks does not change the structure of the TR-VA walker; it just defines the recursive nature of the translations.

## 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.



## 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.

## Global GTT

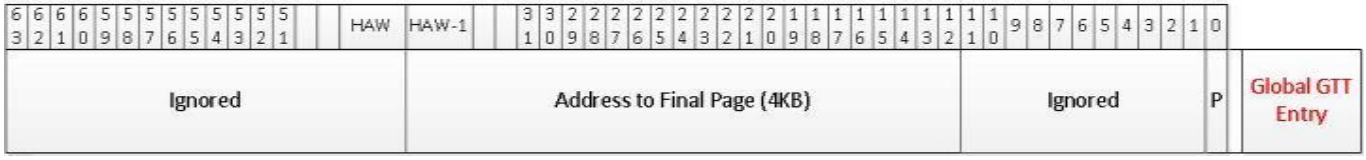
The Global GTT mechanism looks very similar to previous with the distinction of page table entry. Aperture and display will still use the global GTT even if GT core is mapped via per-process GTT.

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:



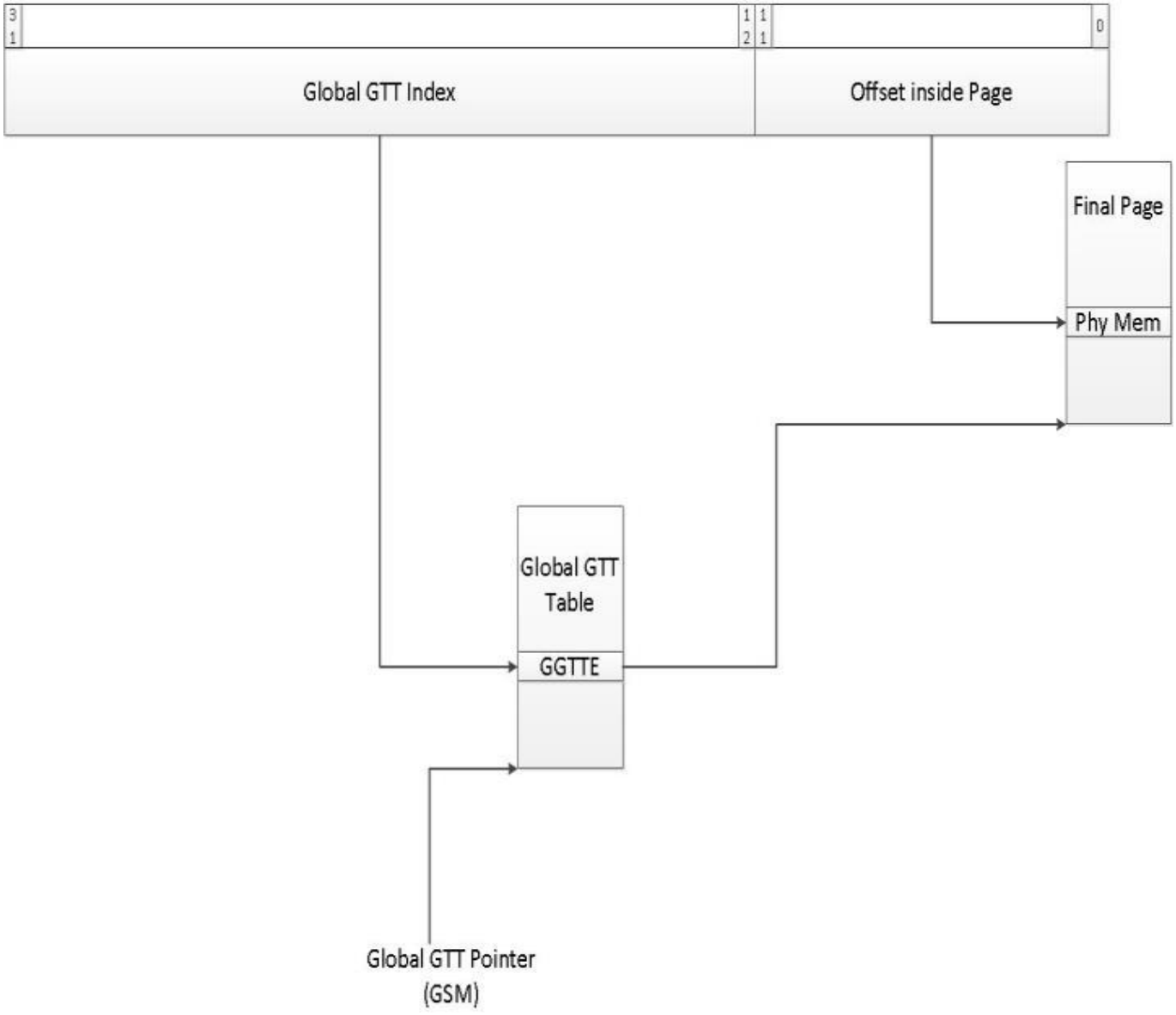

| Bits | Field | Description |
|---|---|---|
| 63:54 | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| 51:HAW | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| (HAW-1):12 | Address | Physical address of 4KB memory page referenced 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. 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.



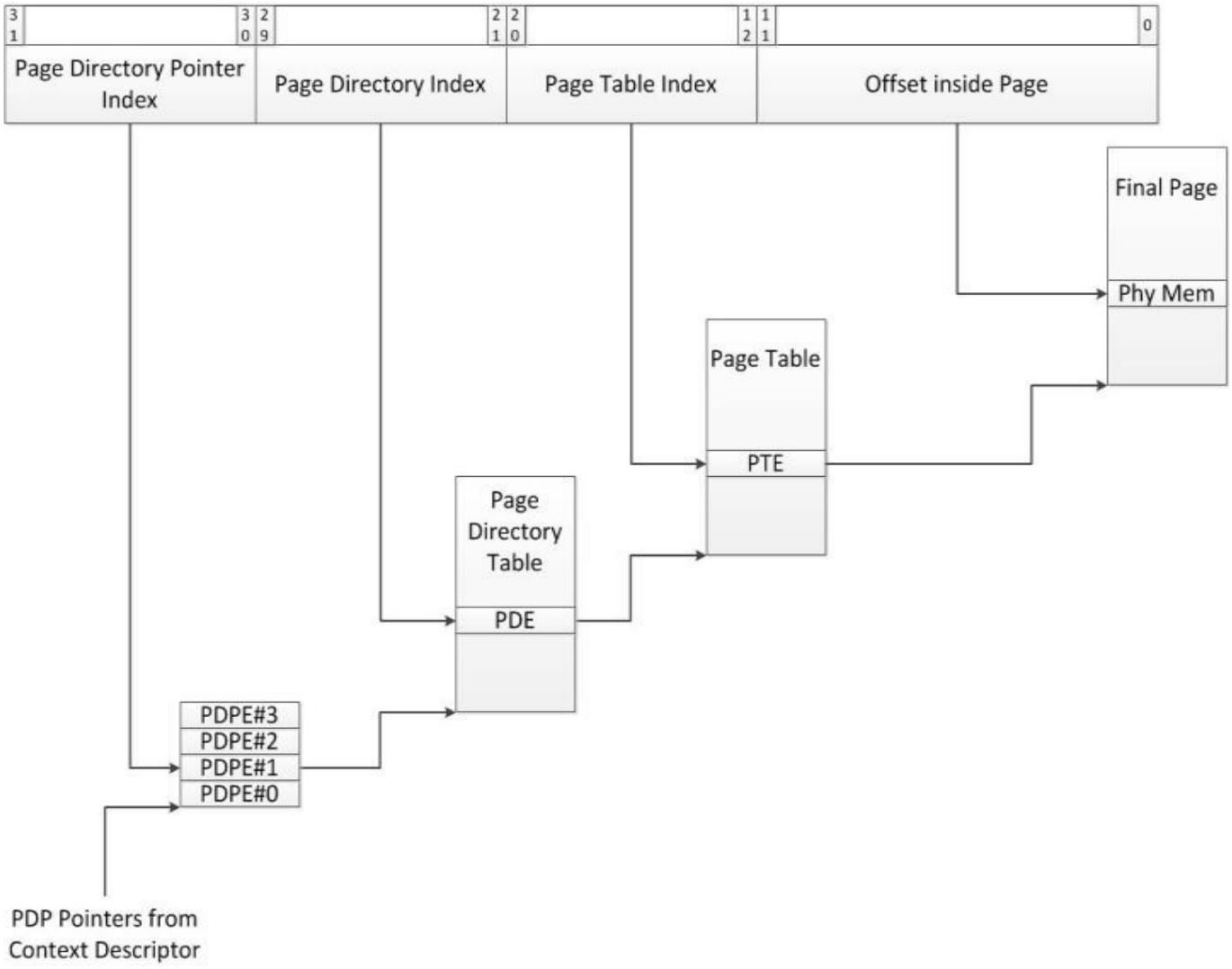

## Legacy mode with 32b VA

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

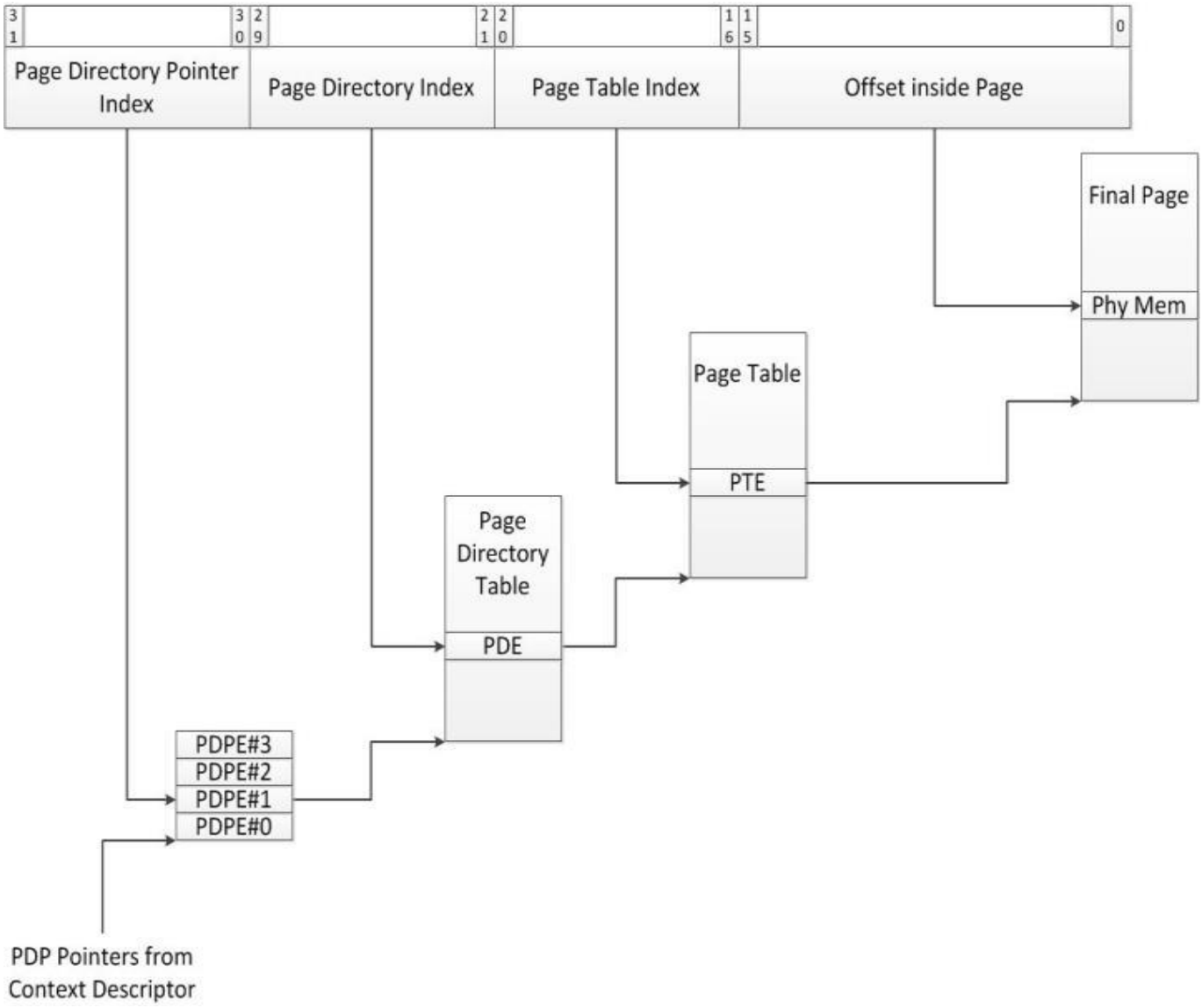
## Page Walk in Legacy mode with 32b VA

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.

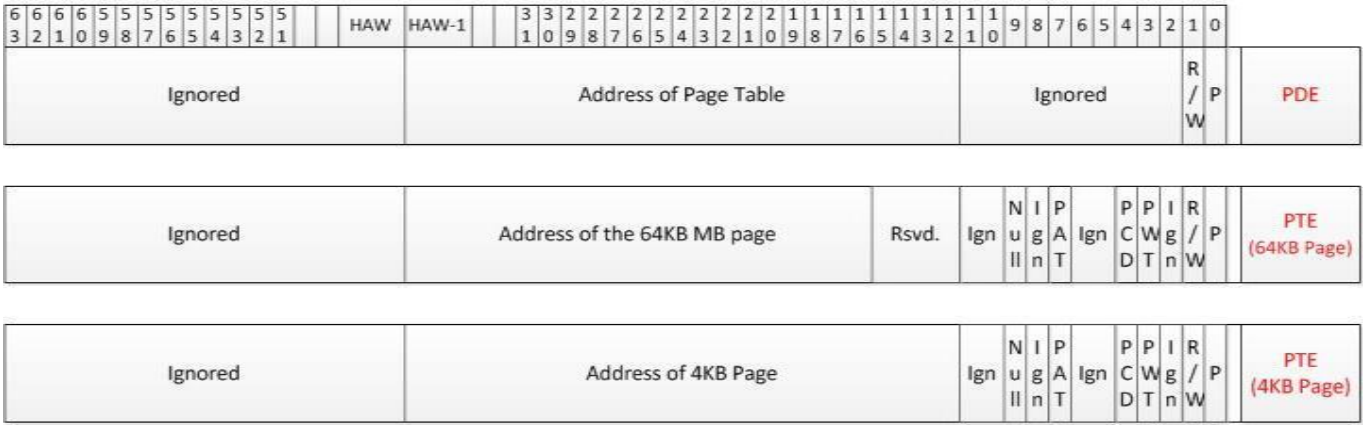

31
30
29
21
20
12
11
0
Page Directory Pointer Index
Page Directory Index
Page Table Index
Offset inside Page
Final Page
Phy Mem
Page Table
PTE
Page Directory Table
PDE
PDPE#3
PDPE#2
PDPE#1
PDPE#0
PDP Pointers from Context Descriptor

## Walk with 64KB Page

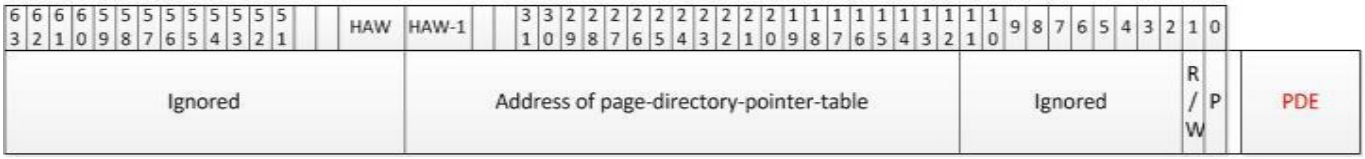


## Page Table Entry (PTE) Formats

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



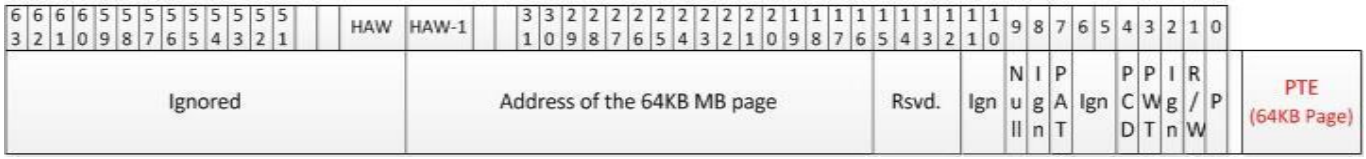

## PDE for Page Table




| 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.<br>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.<br>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.

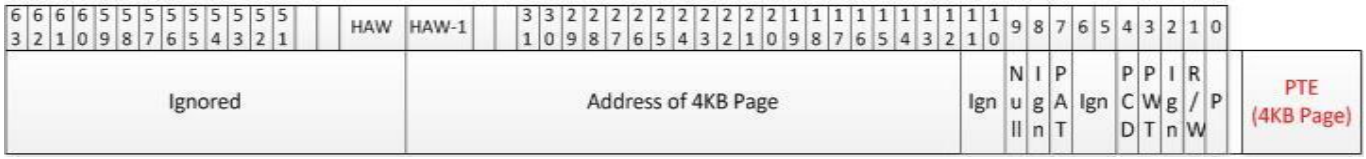
## PTE: Page Table Entry for 64KB Page




| 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.<br>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.<br>*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.<br>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.<br>*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.

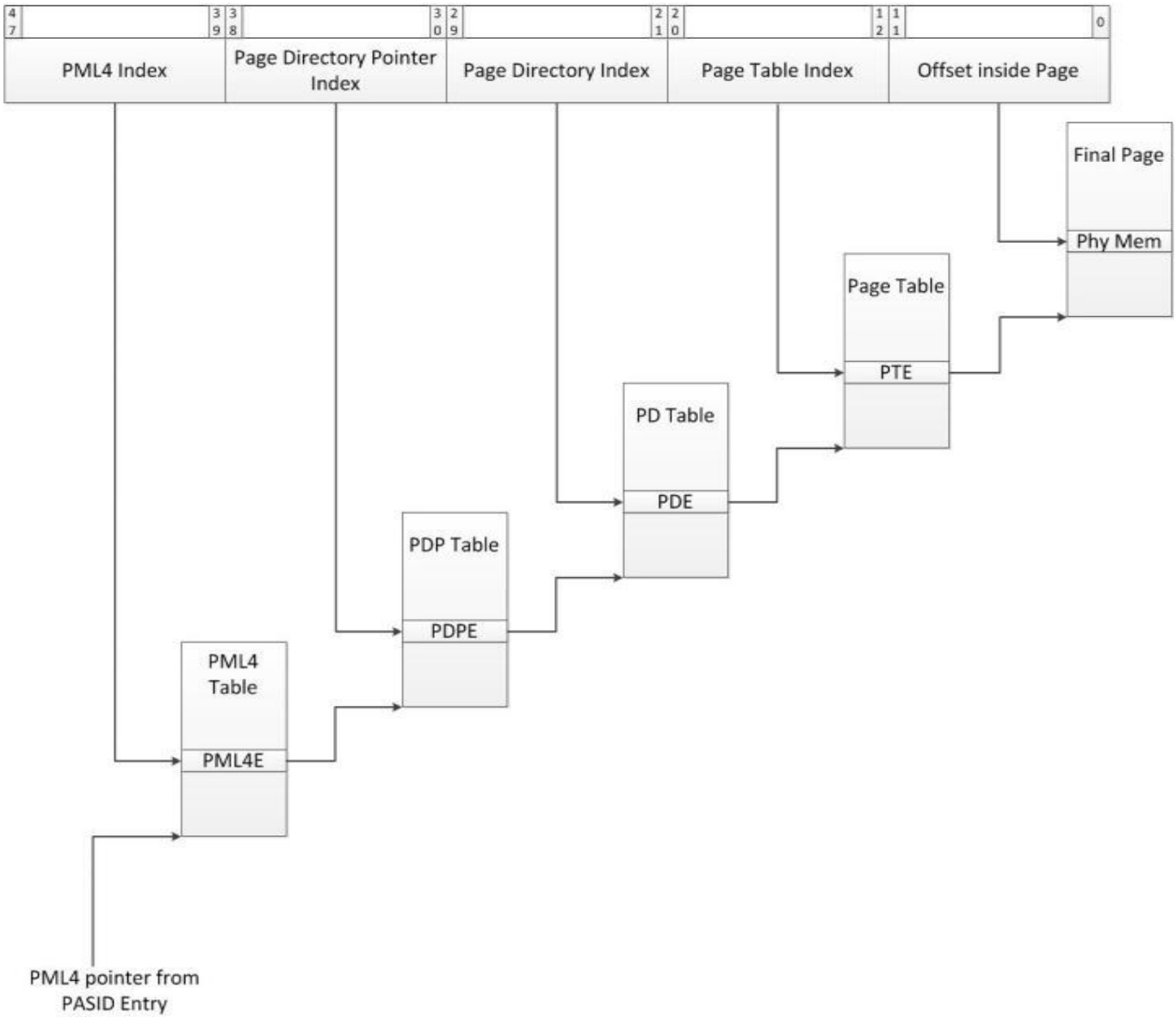
## Legacy mode with 48b VA

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.

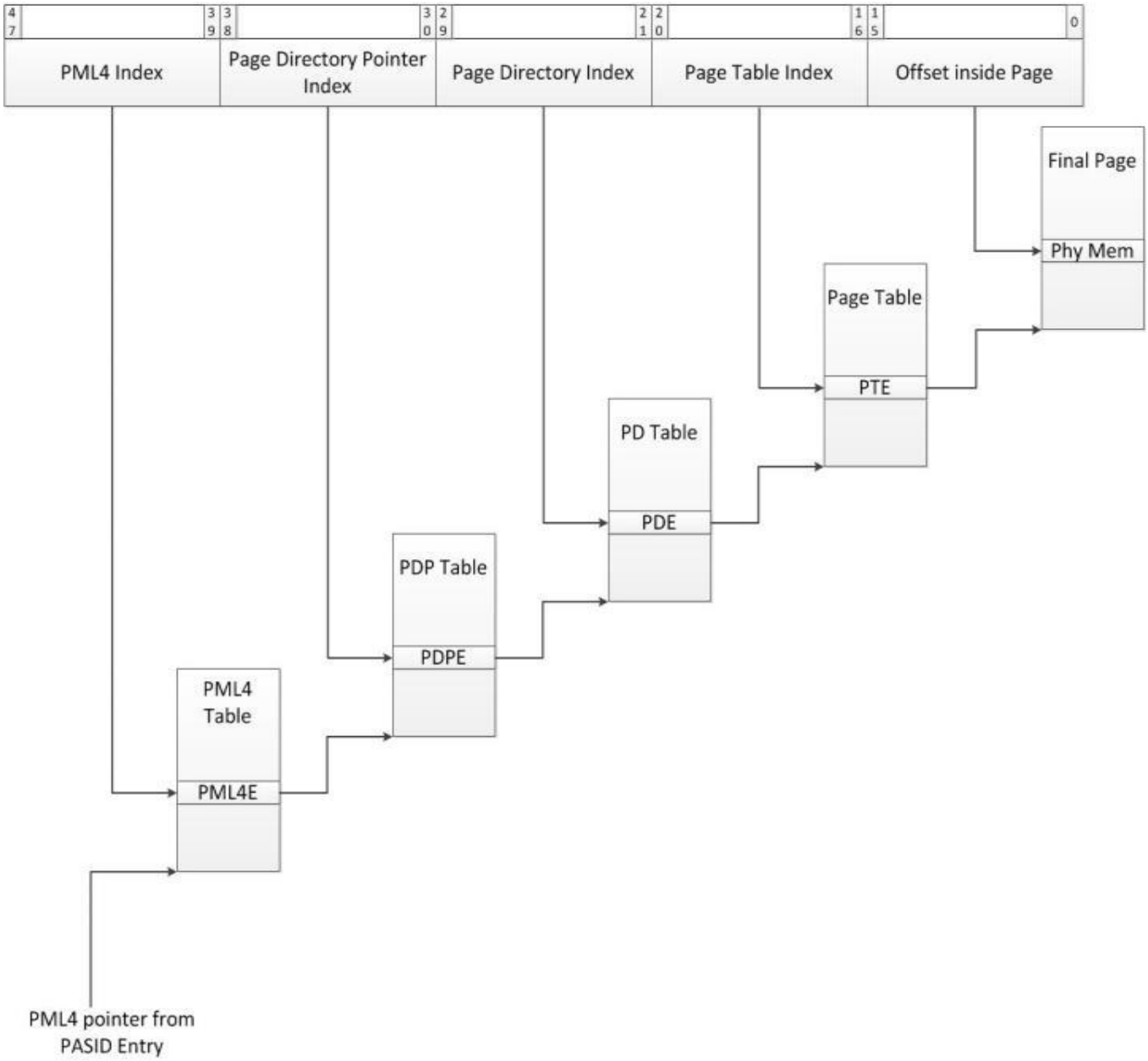
To repurpose the caches the following mechanism will be used:

- 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.

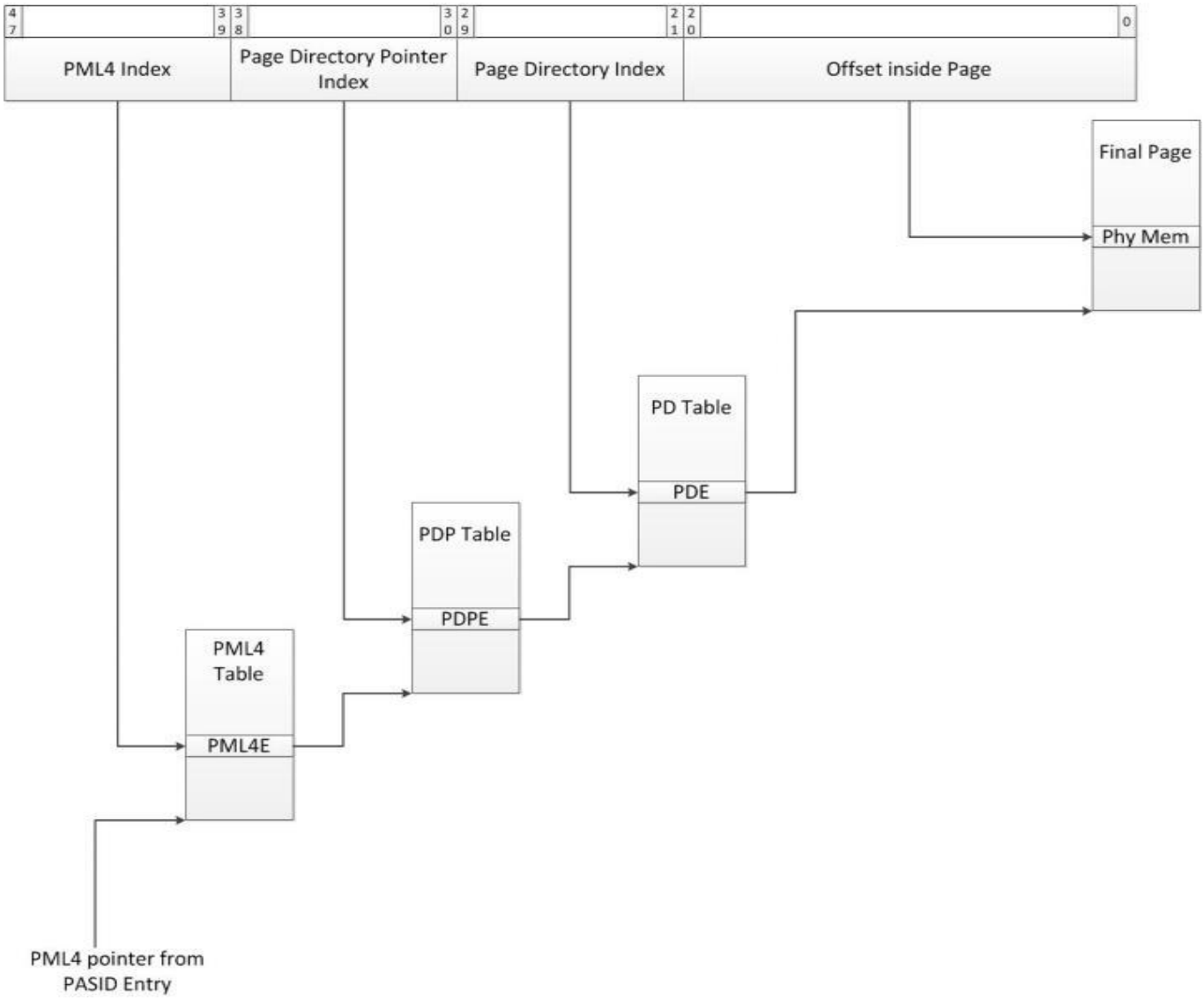
### Walk with 64KB Page

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).



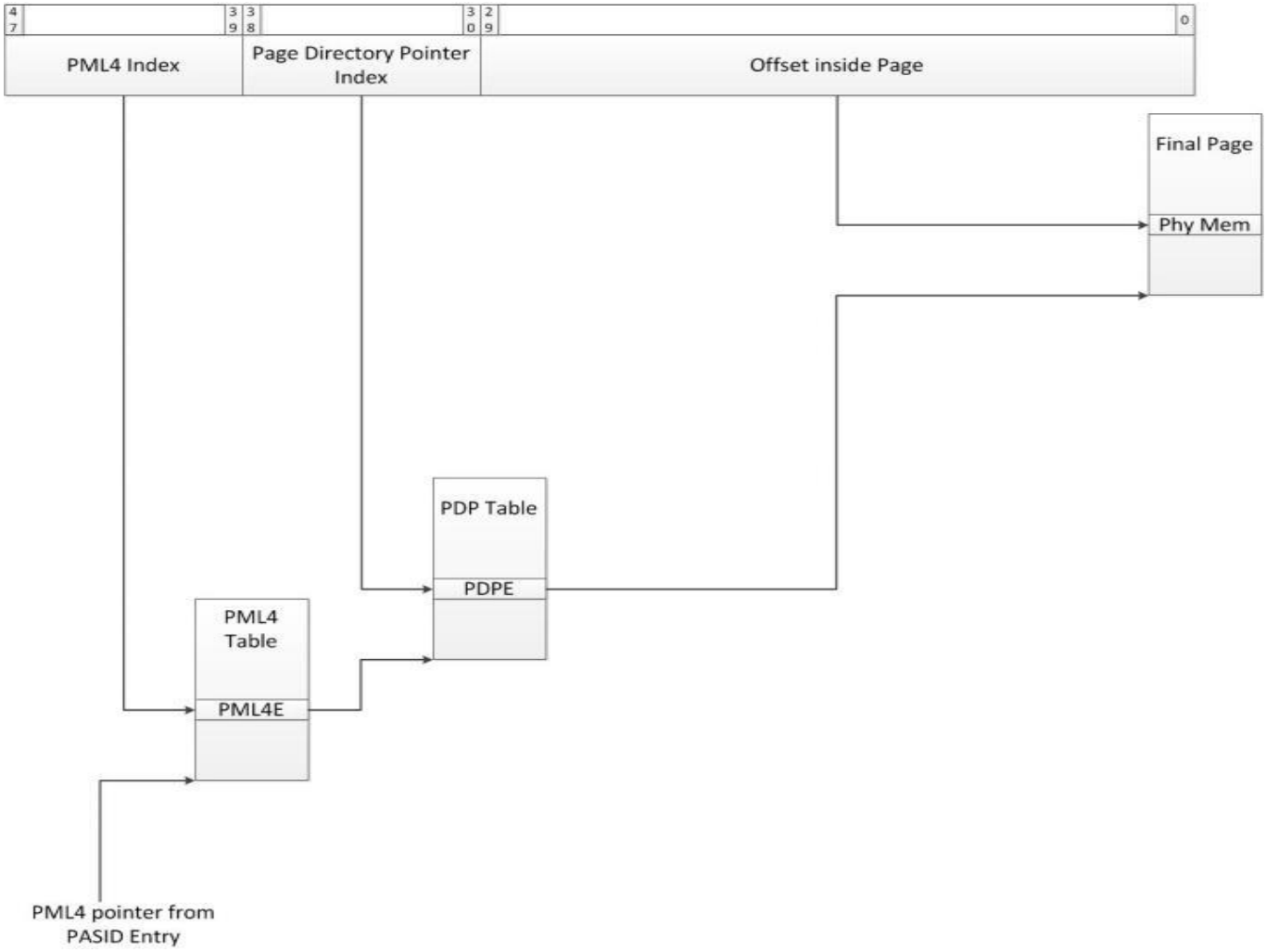
## Walk with 2MB Page

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



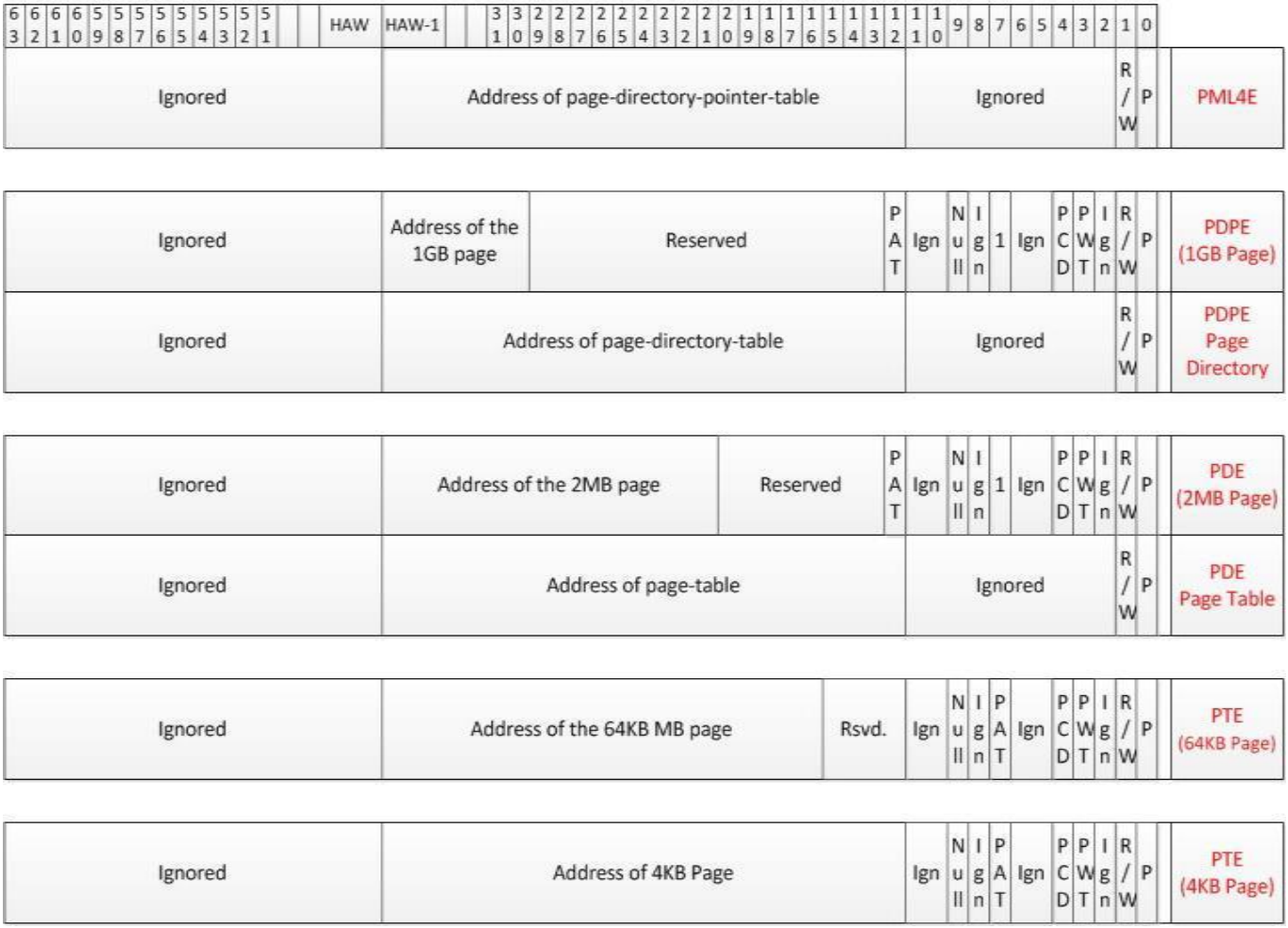
## Walk with 1GB Page

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:

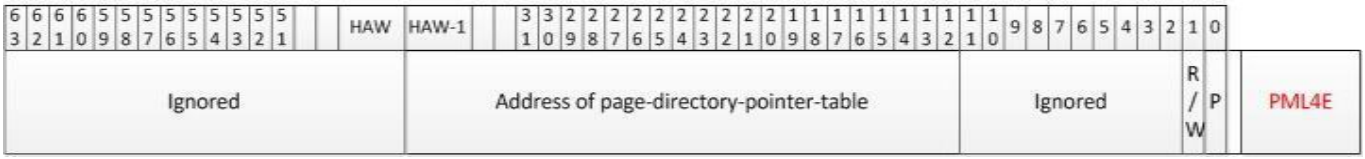



Each table entry is further broken down along with the required functions. GFX has a 4 level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required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

## Pointer to PML4 table

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

## PML4E: Pointer to PDP Table



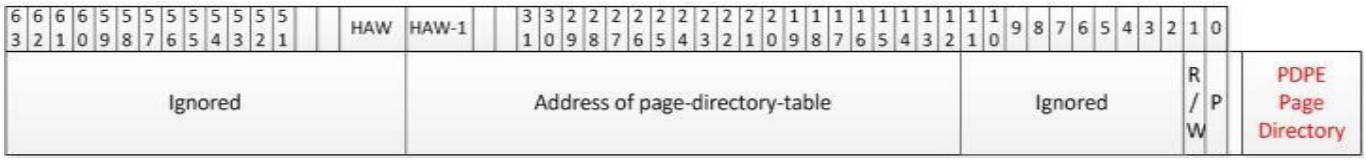

| Bits | Field | Description |
|---|---|---|
| 63:HAW* | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| (HAW-1):12 | ADDR: Address | Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry.<br>This field is treated as Guest Physical 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.<br>*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.

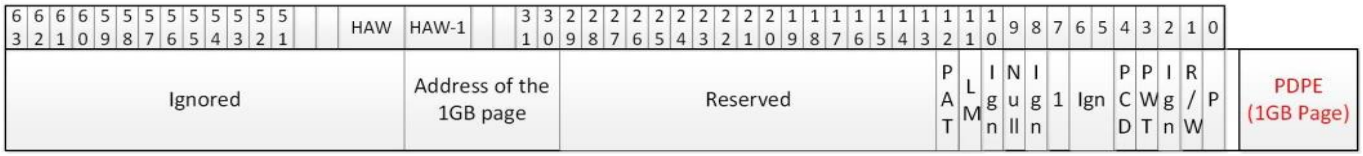
## PDPE for PD




| 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.<br>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.<br>*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.<br>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.<br>*GPU does not support Supervisor mode contexts.* |
| 0 | P: Present | This bit must be "1" to point to a valid Page. |

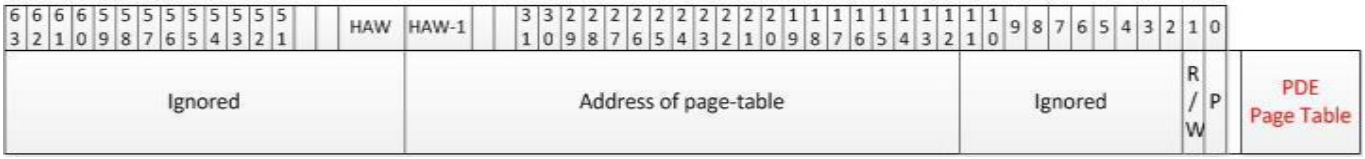
* HAW = 39 for client, and 46 for server.

### 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.<br>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.<br>*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.

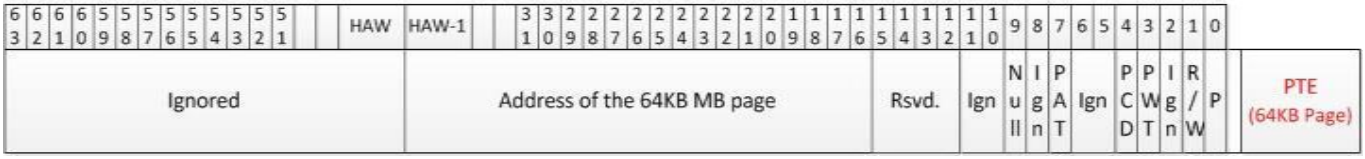
## PDE for 2MB Page


| 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.<br>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)* |
| 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 |
|---|---|---|
| | | *extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights.<br>*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

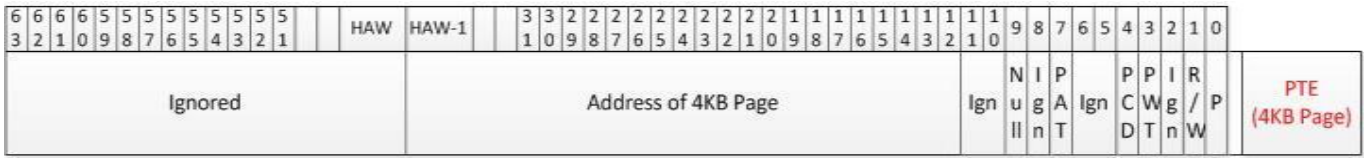



| Bits | Field | Description |
|---|---|---|
| 63:HAW* | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| (HAW-1):16 | ADDR: Address | Physical address of 64KB memory page referenced by this entry.<br>This field is treated as Guest Physical Address (GPA) when Nested 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)* |
| 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 | *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.<br>*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. |

| 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* | *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.

## 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 |
| Per-Process (IA32e) – shared | 4 levels, Page Tables anywhere |

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

## 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

## TLB Caches

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.

### 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. both these have 448 entries.

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

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

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

## Replacement

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

## GTT Walk Request Port (HDC)

A private GTT request port has been added between the HDC(s) and GAM to service the page walks. HDC clusters will contain a mini-TLB and uses GAM's page walker. Their accesses to this page walker is provided thru this private ports. Main GAM TLBs also act as a secondary cache to back these TLBs.

When page walk request comes to GAM, it will be treated as any normal request where the TLB look up will be done and in case of a miss further page walk will be performed. The results of the page walk will be returned on the private connection between the GAM and HDC clusters.

The hierarchy is defined as following diagram where each slice will contain a "Slice GTT Request Manager" (slice GRM) where all HDCs interface with. Each HDC get two credits (i.e. 2-deep ingress queue per HDC) where walk request response back to HDC is considered the release of credits. Slice GRM will collect the walk requests and arbitrate/forward them to GAM on per slice dedicated port.

The request interface is designed to support 1 page walk request per 4 core clocks. Hence both the HDC to slice GRM and slice GRM to GAM should be designed to carry a single page request distributed over 4 clocks to keep the wiring at minimum.

**Page Request Interface:**

- Valid – 1 bit
- Opcode – 1 bit ("0": Page Request and "1": TLB Invalidation Response)
- Slice ID – 1 bit
- HDC ID – 2 bits
- Virtual (GFX) Address – 36 bits (corresponds to [47:12])
- R/W – Read vs Write intend – 1 bit
- Tracking Number – 8 bits
- Faultable vs non-fautable surface - 1 bits

Page response interface from GAM is designed to deliver one-page response every 4 clocks and it is broadcast bus that connects to all HDCs directly. It is up to HDC unit to decode slice/unit ID and claim the response as its own which is also treated as claiming the page miss credit back.

**Page Response Interface:**

- Valid – 1 bit
- Opcode – 1 bit
    - 00: for Page Response
    - 01: Reserved
    - 10: TLB Invalidation Start

- 11: TLB Invalidation End

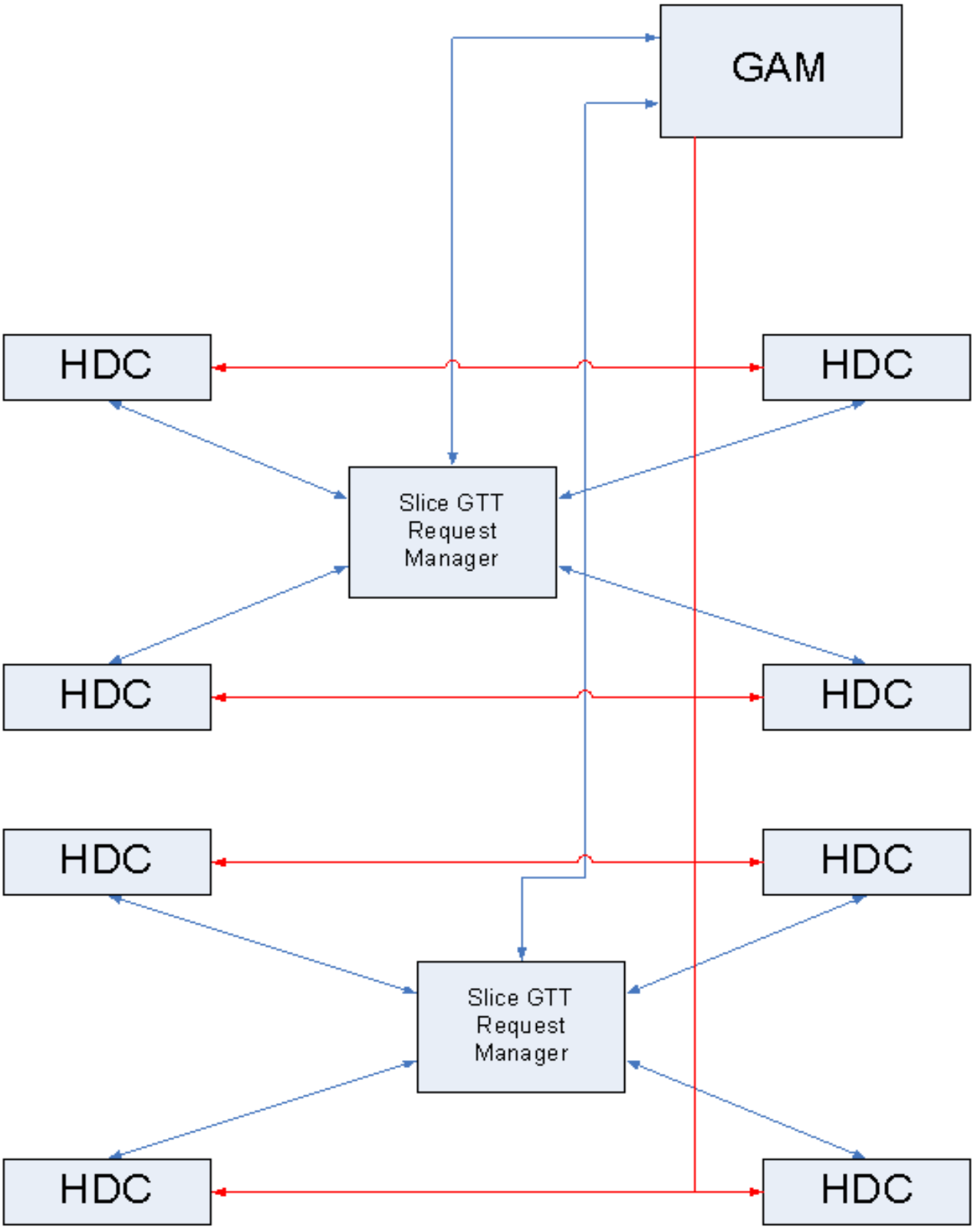
- PA – 27 bits (corresponds to [38:12])
- R/W – this was for a read or write
- Slice ID – 2 bit
- HDC ID – 2 bits
- D bit – 1 bit
- Fault Codes – 2 bits (6 bits)
- Cacheability (memory type) Override – 3 bits
- Tracking number – 8 bits

**Cacheability (memory type) Override** – In case of advanced context execution (where HDC coherent mode is only applicable), the cacheability from surface state will need to be overridden by the OS/VMM setting up the page tables (PAT), MTRR and CD. The effective memory type for HDC has to be used for cache allocation starting with L3. HDC needs to use the memory type bits reported by GAM for memory accesses.

| Memory Type | Encoding in MTRR | HDC to L3 Control[3:2] |
|---|---|---|
| Uncacheable (UC) | 0h | "00" |
| Write Combining (WC) | 1h | "01" |
| Write Through (WT) | 4h | "10" |
| Write Protected (WP) (Reads:WB and Writes:WC) | 5h | Read: "11"<br>Write: "01" |
| WriteBack (WB) | 6h | "11" |
| Reserved* | 2,3,7h | Reserved |

*HDC is already capable of processing WT and WB memory types

**Overall Signaling Diagram for HDC/GAM connection:**



## TLB Invalidation

In addition to page walk requests, there is also a communication needed between HDCs and GAM to relate the TLB invalidation events. GAM combines all TLB invalidation events into a single event as a global TLB invalidation to HDC where the entire content of mini-TLB is wiped out.

The protocol starts with GAM sending a “TLB invalidation start” on *page response* interface. All HDCs will act on the TLB invalidation request as it is a broadcast event. Inline communication of the TLB invalidation is to make sure all previous page responses are seen by the HDCs targeted. Upon receiving the TLB Invalidation start, HDCs will stop sending new TLB requests and only process already available translations pending and ensure corresponding (physical accesses) are GO’ed by L3. Once all these steps are complete HDC will send out the ACK on the “page request” interface to GAM.

GAM will stop sending any page responses post “TLB invalidation start” message and it is free to drop any new request that might have been enqueued by HDCs prior to HDCs seeing the invalidation request. The inline ACK from each HDC is meant to push pending HDC TLB requests towards GAM (where they are dropped). Once GAM collects all “TLB invalidations ACK’s” from all HDCs, it will re-enable the TLB service path and send back (broadcast) “TLB invalidation end” message (inline).

HDCs seeing the “TLB Invalidation end” indicating the sequence are complete and synchronized are free to send TLB requests back to GAM.