# Intel® Arc™ A-Series Graphics and Intel Data Center GPU Flex Series

# Open-Source Programmer's Reference Manual

## For the discrete GPUs code named "Alchemist" and "Arctic Sound-M"

Volume 6: Memory Views

March 2023, 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, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of 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

A modern GPU consists of multiple "engines", including Compute, Render (including Fixed Functions), Media Encode/Decode, Media Enhancement, Blitter/Copy, etc. Engines can operate concurrently and independently using different virtual address spaces.

All engines rely heavily on access to and from memory resources to perform their various functions. The memory subsystem connects engines to the memory resources, and provides services such as address translation, compression, encryption, caching, and HW virtualization. The memory subsystem is the heart of the GPU.

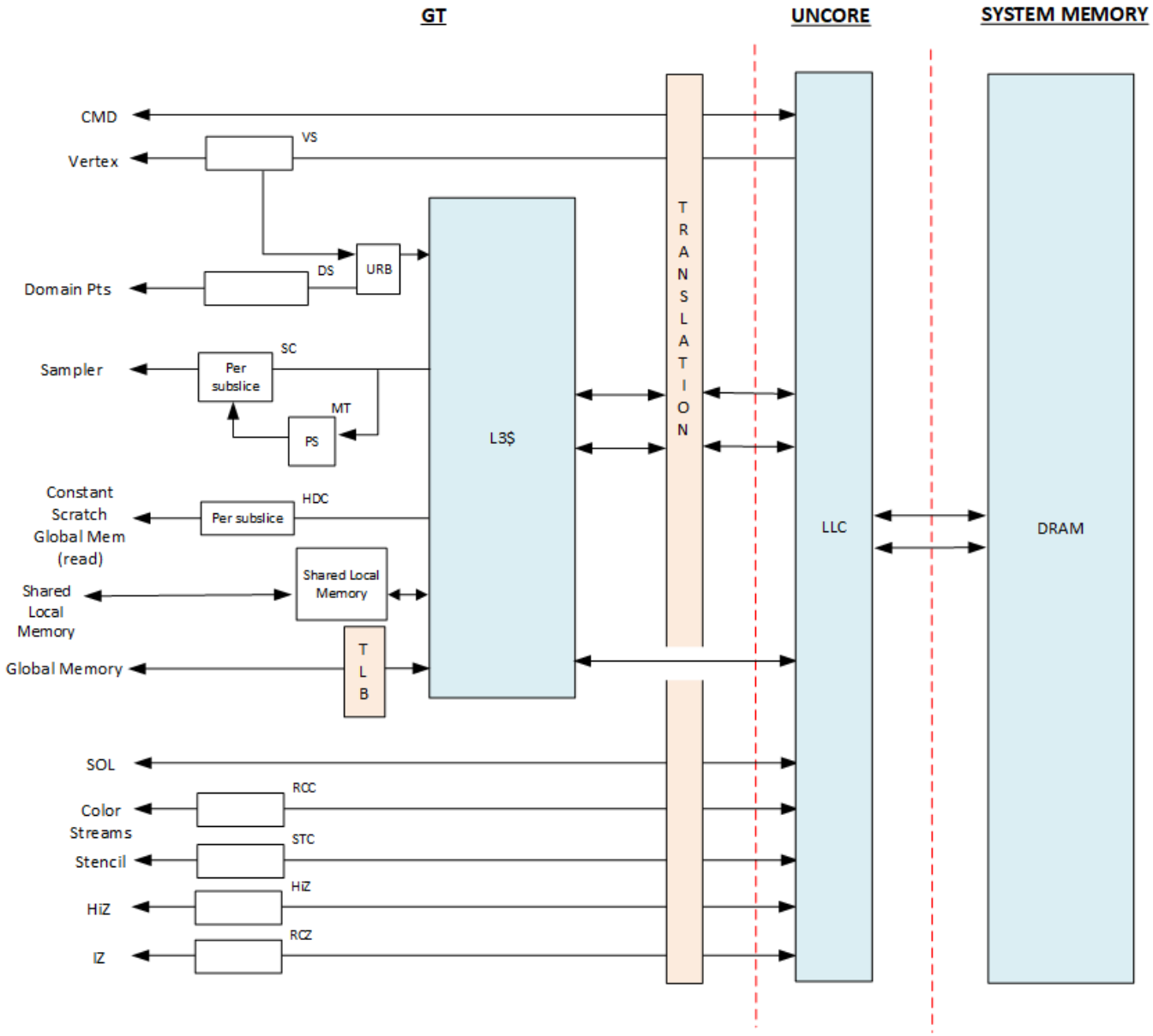
Key components of the memory subsystem include:

- Translation Services: Translation Lookaside Buffer (TLB) that translates virtual addresses associated with a context (process) running on an engine, to the physical address space of the GPU or System, and performs page table walks if a request misses in the TLB
- L3 Cache: Level 3 cache that is shared by all engines within the "GT" domain. Engines may have their own Level 1 and Level 2 caches that are not shared.
- LLC: Last Level Cache of the CPU host (only relevant for Integrated GPU)
- Compression: Handles lossless compression and decompression of memory objects
- Encryption: Handles encryption and decryption of memory objects, potentially including integrity protection, to support various security protocols
- System Memory: Memory that is physically attached to the CPU and managed entirely by the Operating System or Hypervisor
- Device Memory: Memory that is physically attached to a discrete GPU, or "stolen" from System Memory for an Integrated GPU, and managed entirely by the GPU device driver

The memory subsystem supports flows that are coherent with CPU memory (including CPU caches), as well as those that are not coherent with CPU memory. In general, non-coherent flows provide higher bandwidth more efficiently than coherent flows, but may require special handling by SW.

The following diagram provides an overview of a typical memory subsystem an Integrated GPU.

## Cache and Memory Hierarchy



## Graphics Virtual Memory

The GPU uses a virtual memory address space, where the graphics virtual address is mapped through a Page Table 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

The GPU 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) are memory-resident page tables containing an array of Page Translation Entries (PTEs) used in mapping graphics virtual addresses to physical memory addresses. There are two types of page tables: Global GTT and Per-Process GTT.

The base address of the GGTT 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 GGTT size is 8MB, to cover 4GB of Global Virtual Address space, and is physically contiguous (ie, "flat"). The global GTT should only be programmed via the MMIO range within the GTTMMADR BAR. The PPGTT is programmed directly in memory and is multi-level. The page tables are further described in later sections.

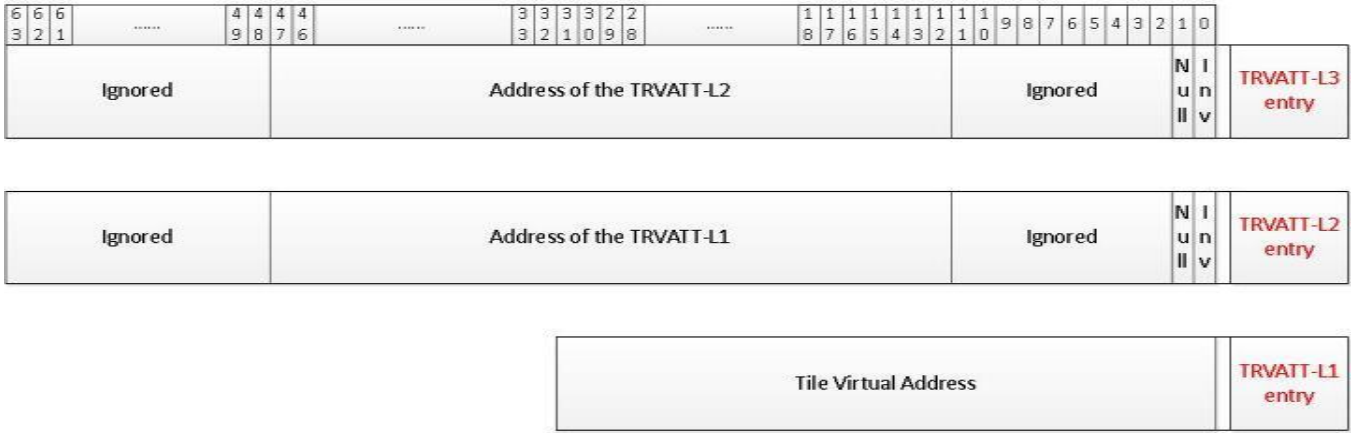
## GFX Page Tables

This section describes the different types of address translation tables used by the GPU.

### 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 is 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. HW will set TRTT tables in Virtual address space mode only.** |
| | 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 zeroes are returned to the originator of the memory access for loads along with a null indicator and for stores the access is dropped at the page walker level.
2. **Invalid Tiles**: This is the case where GFX software did not update the value of the mapping properly for hardware to separate resident vs null tiles. The Invalid Tile treatment is exactly same however additionally a unique interrupt is generated in h/w

Both detections are done by GPU:

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

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

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

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

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

*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.

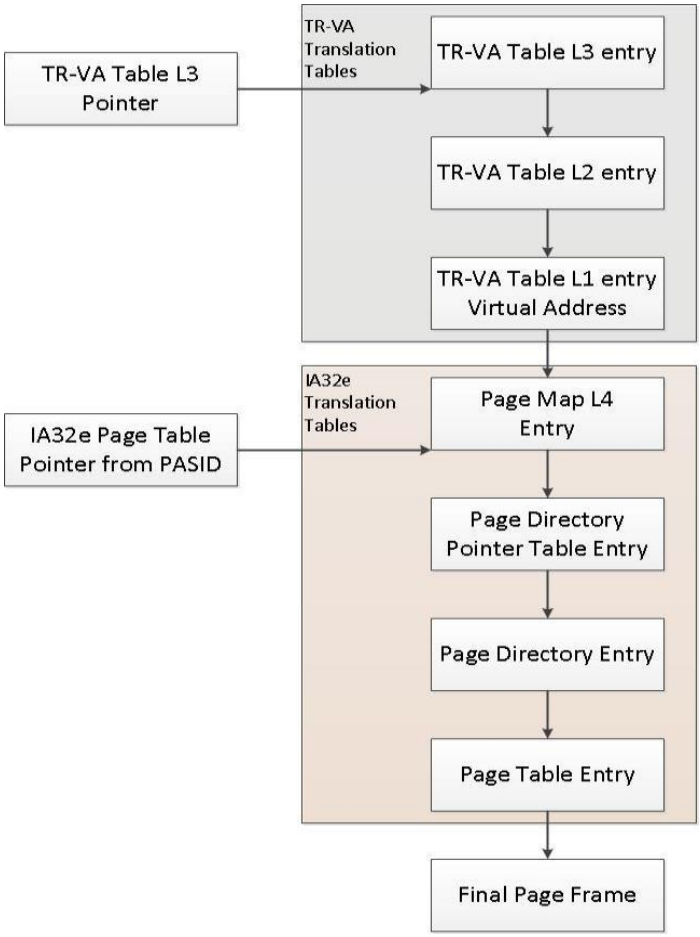
## Physical (Guest) Addressed TR Translation Tables

If the translation tables for TR-VA to VA translation are in physical space, walker does not do 1st level translations for table accesses. Final virtual address will be translated as it is stated in the below diagram, that is no different than any other access that arrives to walker as virtual address to begin with.

To keep the TR-VA translation tables in physical space, GFX software shall ensure that physical mappings behind the corresponding pages are pinned and does not change during the execution of context. It is possible for GFX software to manage pages that contain TR-VA translation tables, however such case requires for GFX software to ensure the consistency of the TLBs and intermediate caches that will contain the table entries. Coherency has to be managed via existing Pipe-Control based TLB invalidation mechanisms.

When physical address pointer is selected, pointer programmed for L3 table has to be within physical memory.

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



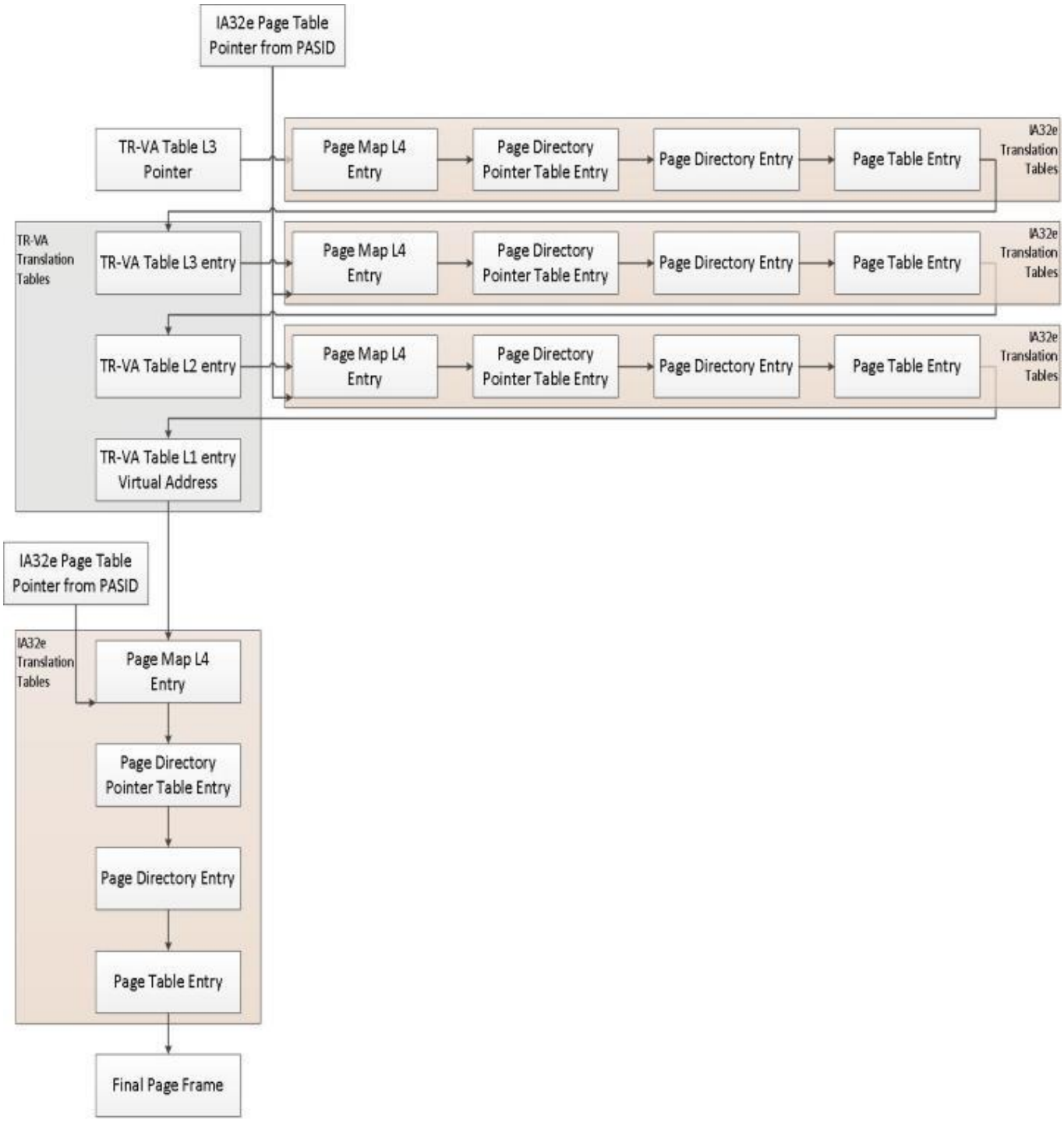

If 2nd Level translations are enabled, all stages have to be walked thru with VTd walker.

*TR translation tables in physical (GPA) address space was implemented due to performance concerns on having them on virtual address space. However, no noticeable degradation has been observed between the two modes. As such, TR translation tables in physical (GPA) address mode is no longer supported.*

## 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.

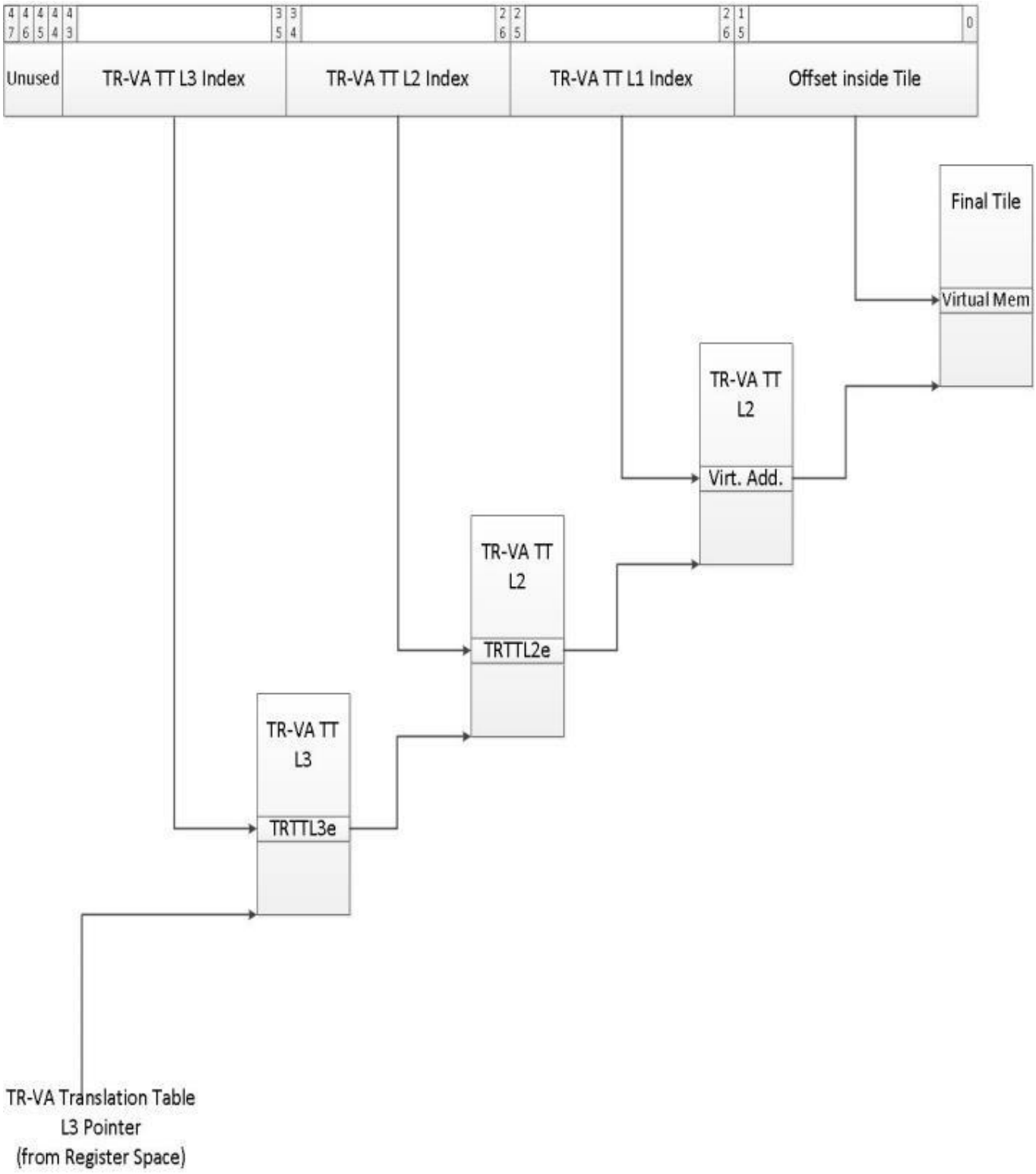
The level of nested walks does not change the structure of the TR-VA walker; it just defines the recursive nature of the translations.

## 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 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.


47 46 45 44 43
35 34
26 25
26 15
0
Unused
TR-VA TT L3 Index
TR-VA TT L2 Index
TR-VA TT L1 Index
Offset inside Tile
Final Tile
Virtual Mem
TR-VA TT L2
Virt. Add.
TR-VA TT L2
TRTTL2e
TR-VA TT L3
TRTTL3e
TR-VA Translation Table L3 Pointer (from Register Space)

## 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 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

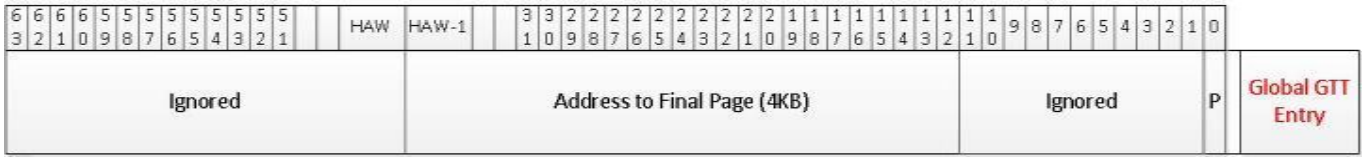
The GPU supports a Global Virtual Address space that is used for resources that are accessible to privileged (ie kernel-mode) processes, and not tied to a specific user-level process. For example, the Graphics micro-Controller (GuC) and Display Engine (if present) utilize Global address space. The Global VA space is 4GB in size (32-bit VA).

The Global GTT (GGTT) translates from the Global virtual address to a physical address that can be accessed by HW, using a page size of 4KB. The GGTT is a flat, single-level table. Each page table entry (PTE) in the GGTT is 8B in size, so the overall size of the GGTT 8MB.

The Global GGTT entries can be updated by the graphics driver, via the GTTMMADR MMIO range of the device, and also via the Command Streamer command MI_UPDATE_GTT.

## Page Table Entry

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




| 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. |
| 11:2 | 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. The Function number must be less than or equal to the value of the Num VF register in the SRIOV Capability structure. |

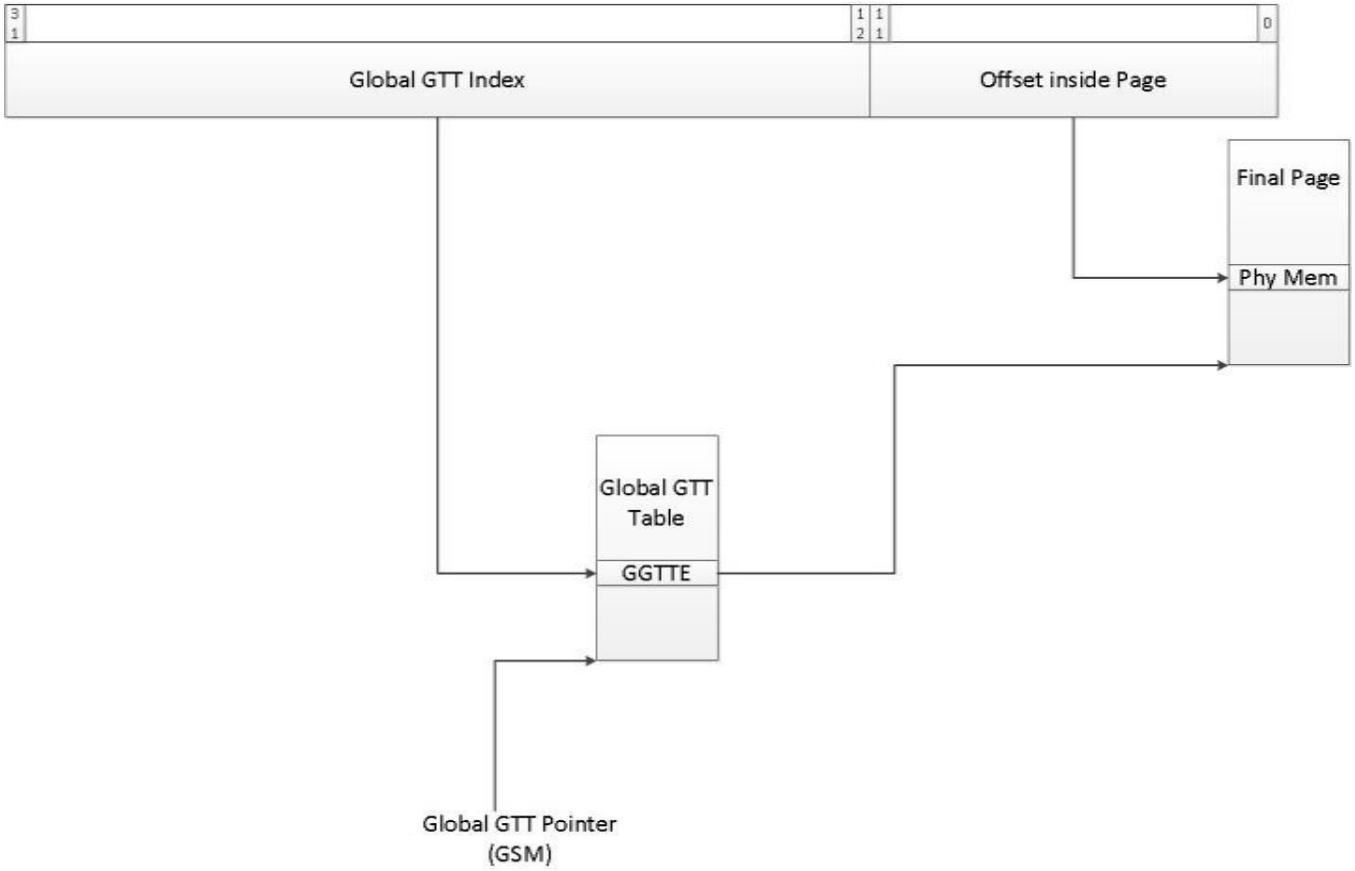
| Bits | Field | Description |
|---|---|---|
| 1 | Local Memory | Indicates the page is allocated in Local Memory instead of System Memory. This field is ignored in configurations that do not allocate Local Memory.<br>When Global Address is mapped to a large page (>4k), multiple consecutive GGTT entries, with the same attributes in each, must be used to cover the large page. For example: a 64KB page must be programmed as 16 consecutive 4KB pages with LM='1 and the same Function number on all. |
| 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 previous versions. 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 48b VA

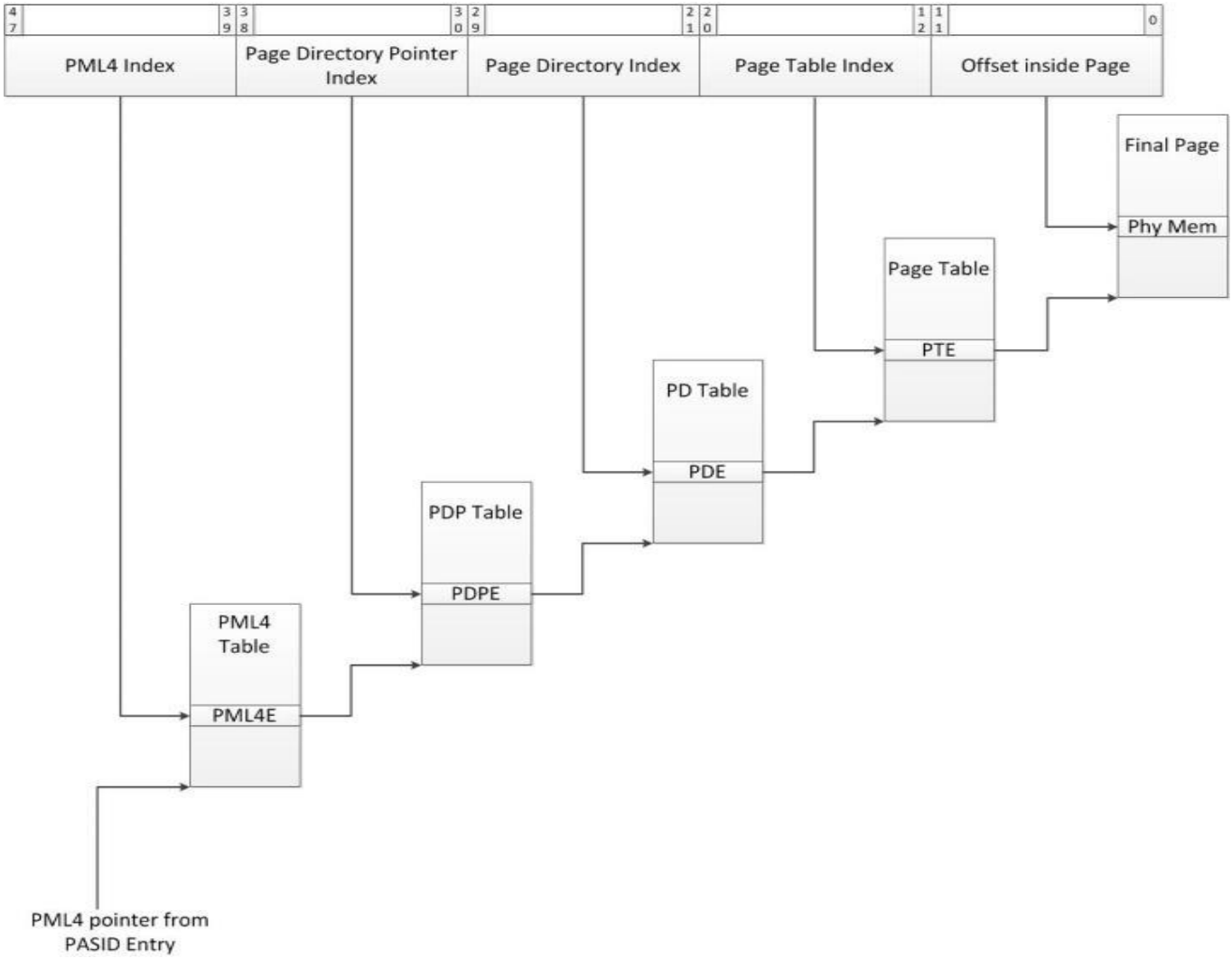
The GPU typically operates on behalf of user-level processes (applications), each of which has it's own "Per-Process" virtual address space. The size of this space is 256TB (48b address width).

The Per-Process GTT (PPGTT) translates these virtual addresses to physical addresses that are used by HW. The PPGTT is a multi-level table very similar to the IA32e table utilized by Intel CPUs. Each entry in the PPGTT is 8 Bytes.

## Page Walk in Legacy 48b Mode

Translation of a 48b VA requires 4 levels of page table. assuming each table level is 4KB and each entry is 8B. The top-most level of the PPGTT is located via the PML4 table pointer associated with the process, and the 48b VA as used to index into consecutive levels of page table.

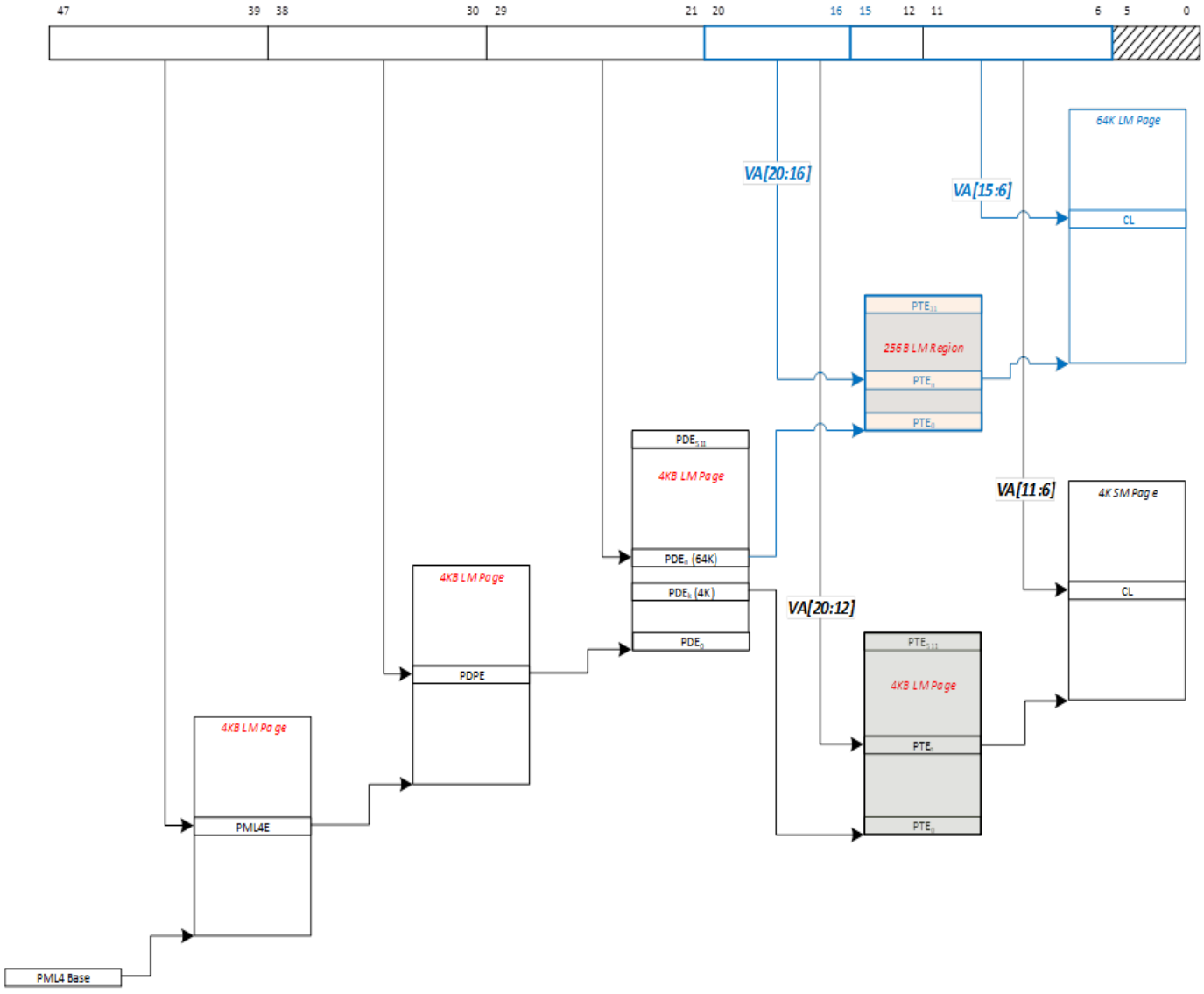
The following diagram shows the page walk that is needed for a 4KB page.



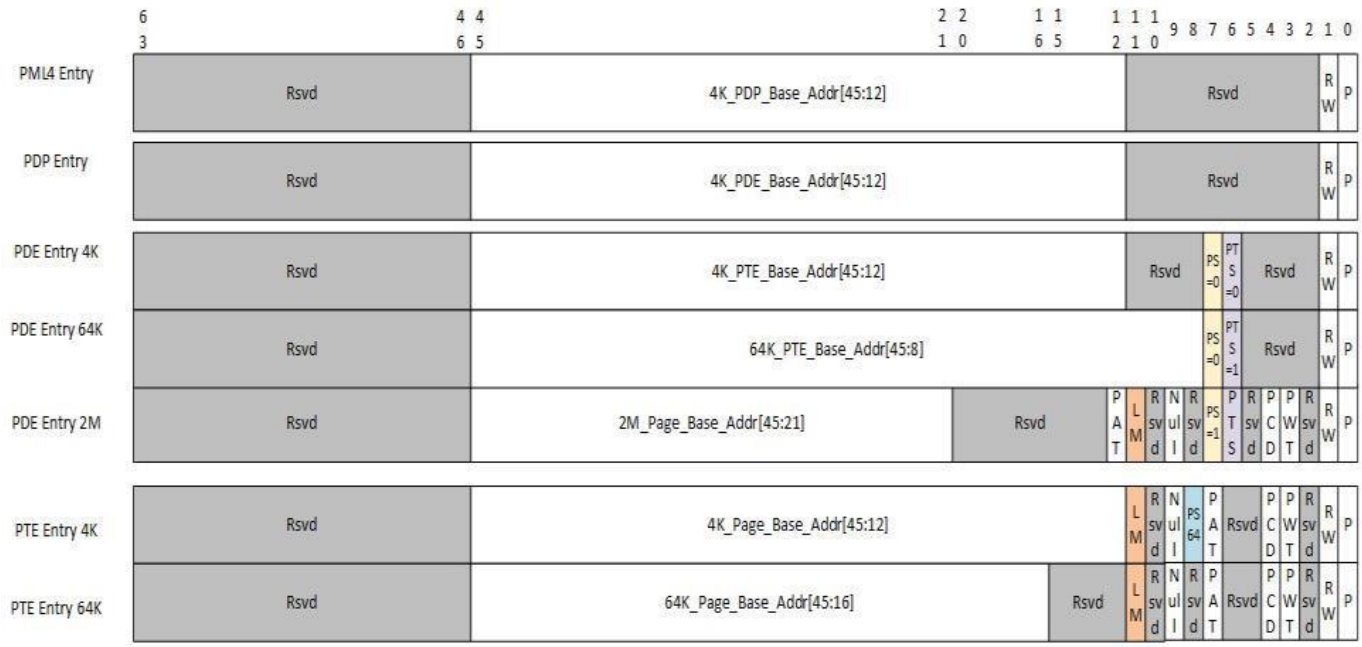
## Walk with 64KB Page

Page table structure for 64K pages is optimized for Local Memory allocations. 64K page indication still lies in the Page Directory Entry. Each PDE still covers 2M space underneath. Thus, when a PDE describes 64K pages underneath only 32 PTEs are needed to map the 32, 64K pages within the 2M space. In the case of 64K PDE, the PDE points to a 256B space that holds the 32 PTEs.

The Page Table and Page Walk structure is as shown below.




The structure and the contents of each table entry is shown below.




First 2 levels of the walk (PML4 and PDPE) remains the same. Page Directory table structure also remains the same, but the contents of the Page Directory Entry (PDE) are changed.

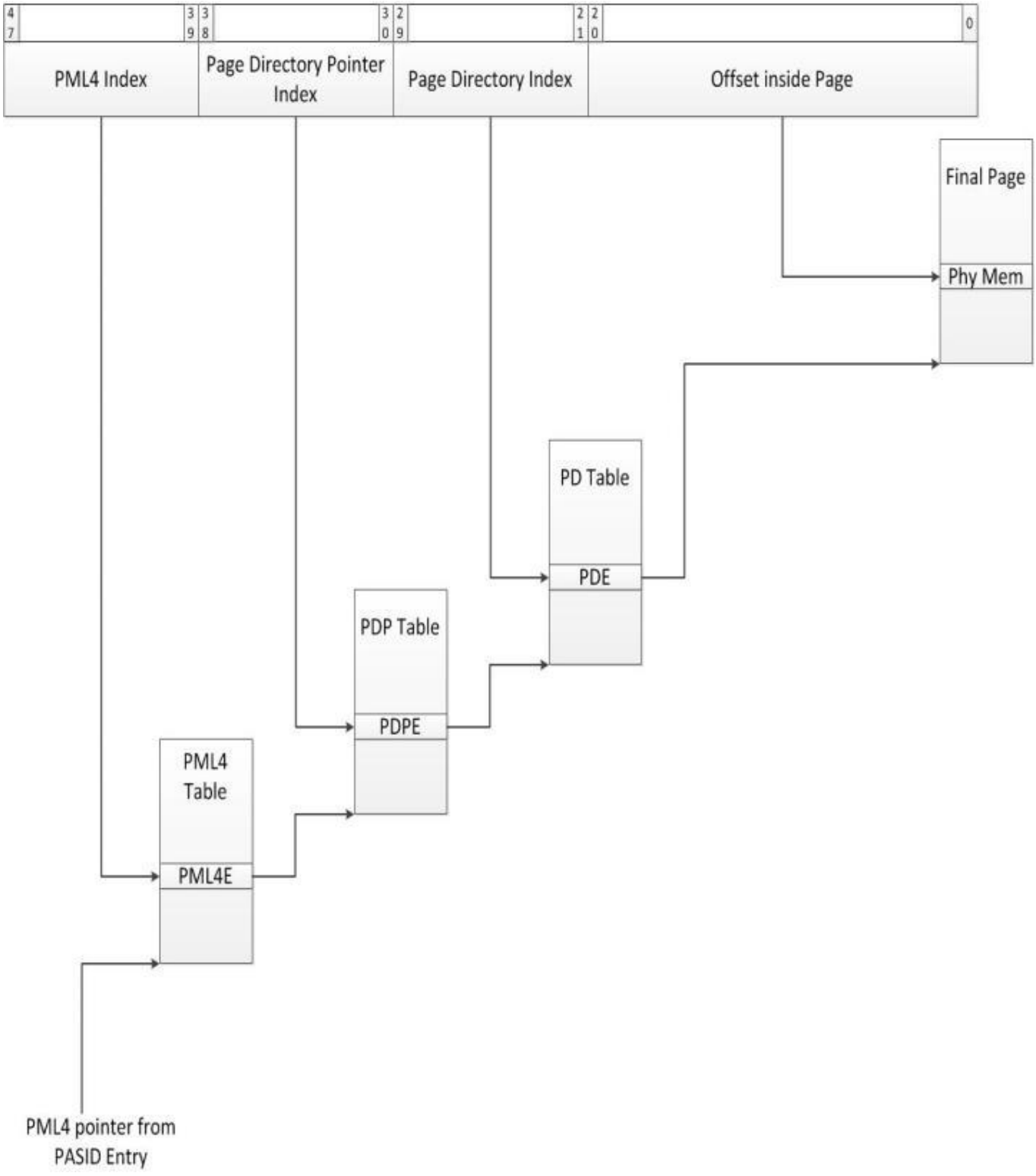
- PDE entry gives a 4KB aligned pointer ([45:12]) to the PTE for 4KB physical page allocations, and a 256B aligned pointer ([45:8]) to the PTE for 64KB physical page allocations.
- 4KB/64KB PTE indication is moved from PDE[11] to PDE[6]

When the underlying physical pages are 4KB each, a 4KB Page Table is used, which holds 512 Page Table Entries (4K PTEs). When the underlying physical pages are 64KB each, a 256B Page Table is used which holds 32 Page Table Entries (64K PTEs). The leaf level entries (PTE for 4/64K and PDE for 2M) also hold the Local Memory (LM) bit. This bit indicates the resulting GPA resides in the Local Memory or System Memory.

Discrete GPUs do not support 4K pages to be allocated in Local Memory. Therefore, when PPGTT.PTE[11] = '1, indicating the page is in Local memory, PPGTT.PDE[6] must be set to '1, indicating it is a 64K page.

## 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.

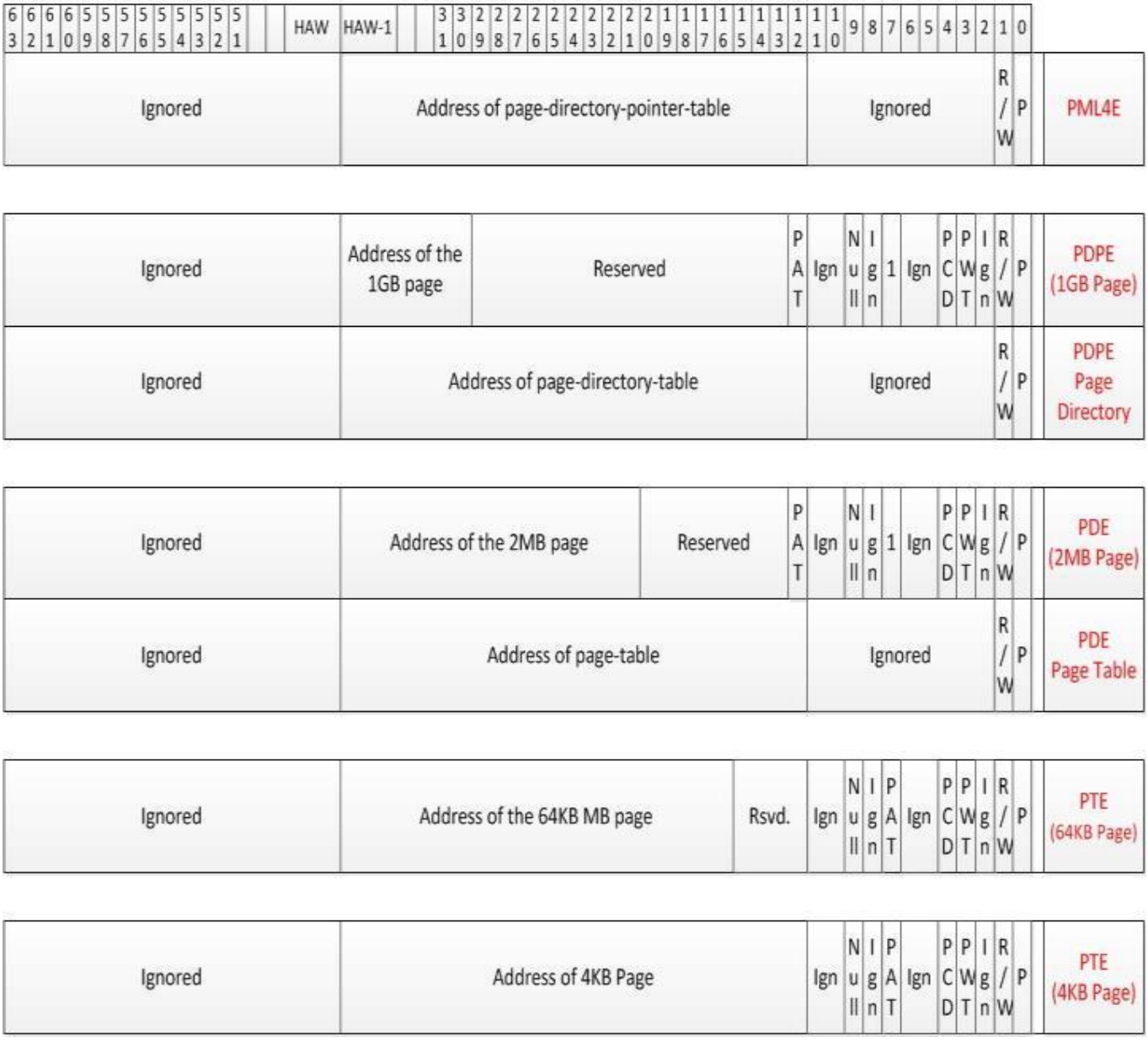



## Walk with 1GB Page

1GB pages are supported by completing address translation at the PDP level, similar to how 2MB page translation is complete at PDE level.

## Page Tables Entry PTE Formats

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

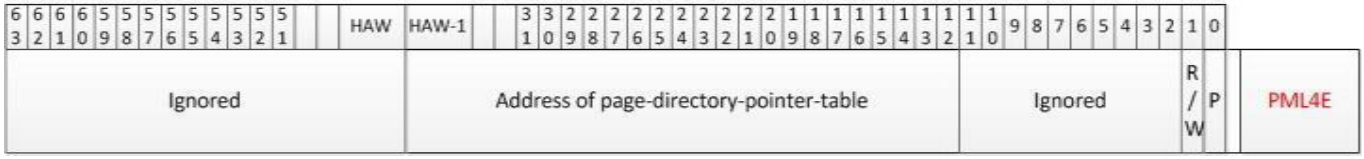



Each table entry is further broken down along with the required functions. GFX has a 4 level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required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

## Pointer to PML4 table

The pointer to PML4 table is provided via the context descriptor.

## PML4E: Pointer to PDP Table



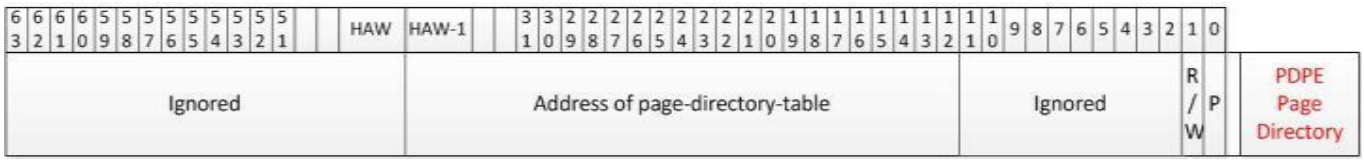

| Bits | Field | Description |
|---|---|---|
| 63:HAW* | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| (HAW-1):12 | ADDR: Address | Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry.<br>This field is treated as Guest Physical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

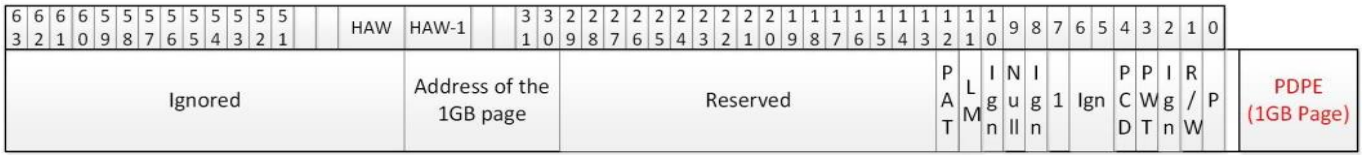



| Bits | Field | Description |
|---|---|---|
| 63:HAW* | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| (HAW-1):12 | ADDR: Address | Physical address of 4-KByte aligned page-directory table referenced by this entry.<br>This field is treated as Guest Physical Address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

| Bits | Field | Description |
|---|---|---|
| | | (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 zeros for the read access with a null completion, write accesses are dropped. |
| 8 | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| 7 | PAT: Page Attribute | For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. |

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

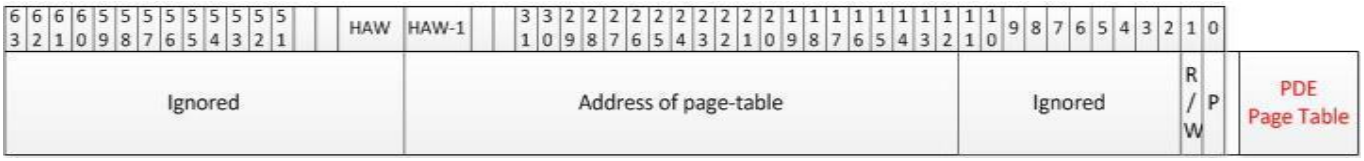
* HAW = 39 for client, and 46 for server.

## 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)* |

| Bits | Field | Description |
|---|---|---|
| 1 | R/W: Read/Write | Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights.<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.

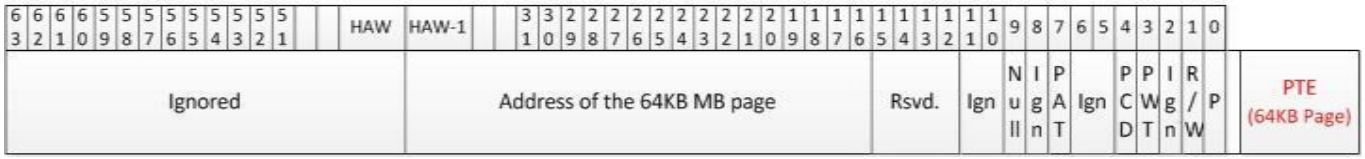
## PDE for 2MB Page


| 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 zeros for the read access with a null completion, write accesses are dropped. |
| 8 | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| 7 | PAT: Page Attribute | For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry. |
| 6:5 | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |

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

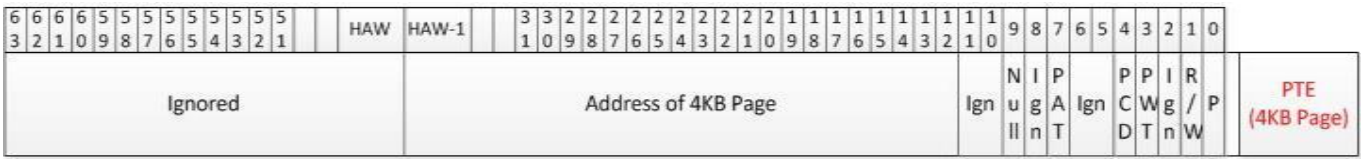



| Bits | Field | Description |
|---|---|---|
| 63:HAW* | *Ignored* | *Ignored (h/w does not care about values behind ignored registers)* |
| (HAW-1):16 | ADDR: Address | Physical address of 64KB memory page referenced by this entry.<br>This field is treated as Guest Physical Address (GPA) when Nested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

| Bits | Field | Description |
|---|---|---|
| | | 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 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 | 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 |

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

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

## Complete Page Walk

There are various types of page walks which are dependent on which context GPU is running as well as whether the VTd translations (2nd level page tables) are enabled or not.

## Legacy Context

Legacy context could use either Global GTT or Per Process GTT which is given to page walker as part of the context descriptor. Even under PPGTT, there could be accesses from Command Streamers that would require using Global GTT, which requires to treat the walk requirement per transaction.

Under PPGTT, there could also be accesses from u-controller that would require using Global GTT, which requires to treat the walk requirement per transaction.

In addition, VTd translations (2nd level Page tables) can be dynamically enabled or disabled via virtual machine monitor which would require the VTd walker to come in between the 1st level page walker and memory access ports.

For Legacy context indicator command streamer is going to pass the "context type" information along with other parameters that defines how certain behavior for paging needs to be.