Memory Protection

Lecture 9

Memory Protection

- Memory Protection Unit
- Memory Management Unit

Memory Protection

ARM: MPU, RISC-V: PMP

Memory Management

MMU

Bibliography

for this section

1. **Andrew Tanenbaum**, *Modern Operating Systems (4th edition)*
   - Chapter 3 - *Memory Management*
     - Subchapter 3.3 - *Virtual Memory*
2. **Philipp Oppermann**, *Writing an OS in Rust*
   - *Introduction to Paging*
   - *Paging Implementation*

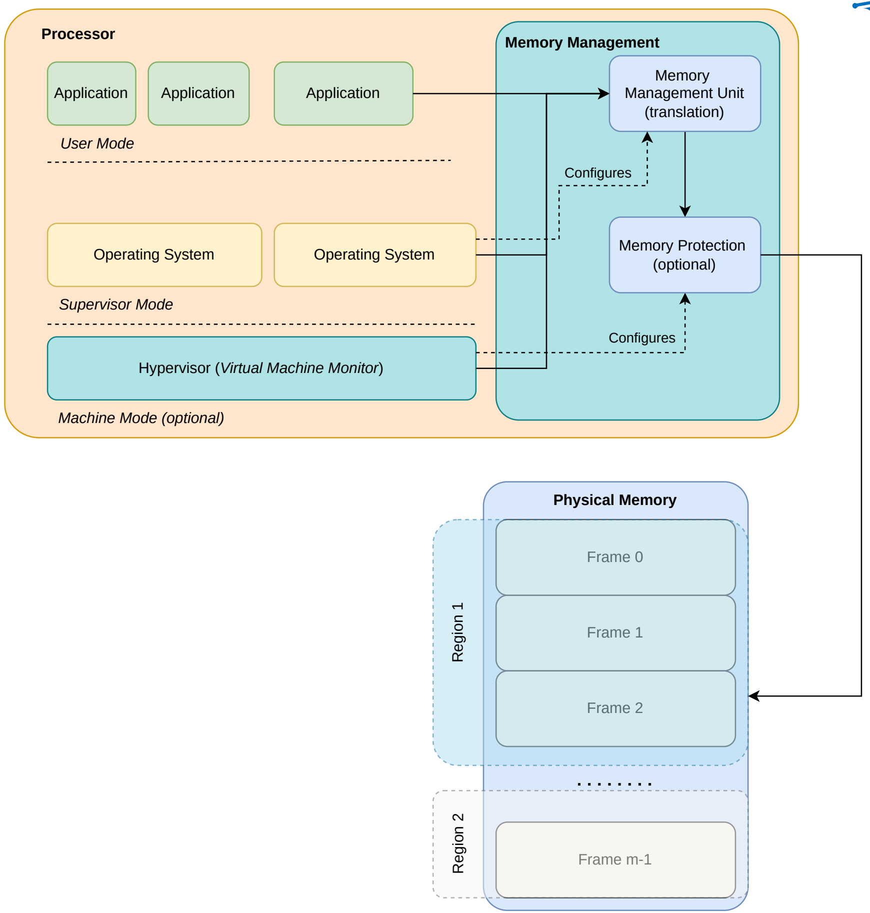
Memory Management

memory access defined page by page

- uses *logical addresses*
- **translates** to *physical addresses*

The processor works in at least two modes:

- **supervisor** mode
  - restricts access to some registers
  - accesses virtual addresses through Memory Protection (*if machine mode exists*)
- **user** mode
  - allows only ALU and memory load and store
  - accesses memory access through the Memory Management Unit (*MMU*)



Paging

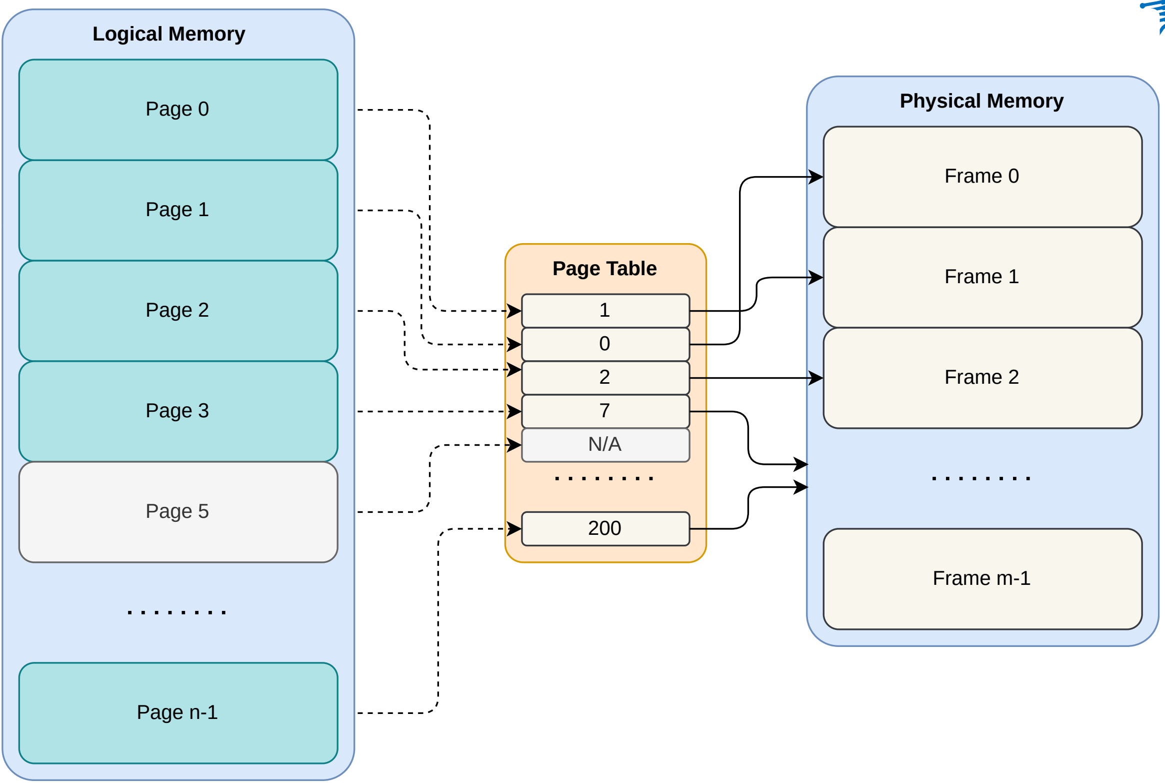
the memory *unit* is the page

- Physical Memory (*RAM*) is divided in **frames**
- Logical Memory is divided in **pages**
- *page* = *frame* = **4 KB** (usually)

logical addresses are translated to *physical addresses* using a **page table**

the **page table** is located in the **physical memory**

- each memory access requires at least 2 memory accesses



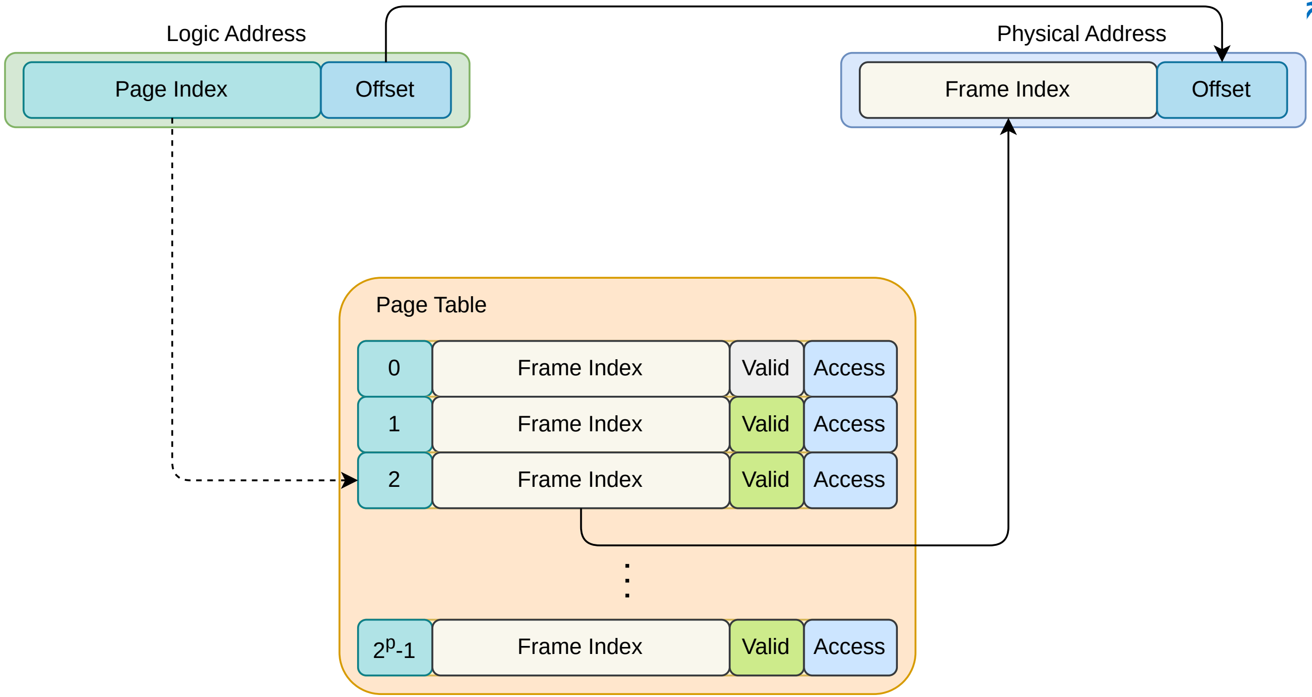
Address Translation

page to frame

the logic address is divided in two parts:

- *page index*
- *offset* within the page

the MMU translates every logic address into a physical address using a *page table*



Logic Address

Page Index | Offset

Physical Address

Frame Index | Offset

Page Table

0	Frame Index	Valid	Access
1	Frame Index	Valid	Access
2	Frame Index	Valid	Access
⋮			
2^p-1	Frame Index	Valid	Access

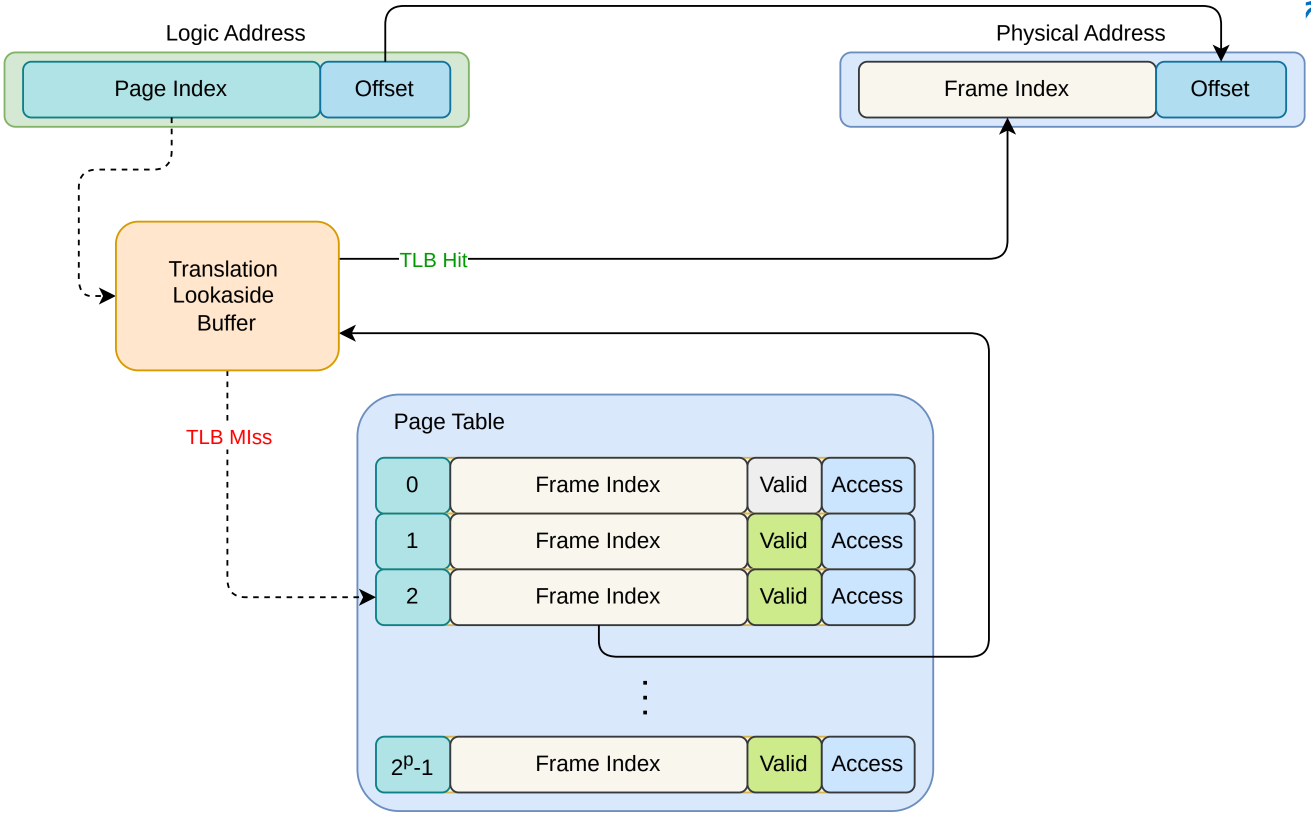
Translation Lookaside Buffer (TLB)

caching address translation

the **page table** is **stored in RAM**

each memory access **requires 2 accesses**

1. read the page table entry to translate the address
2. the requested access



Page Directory

caching address translation

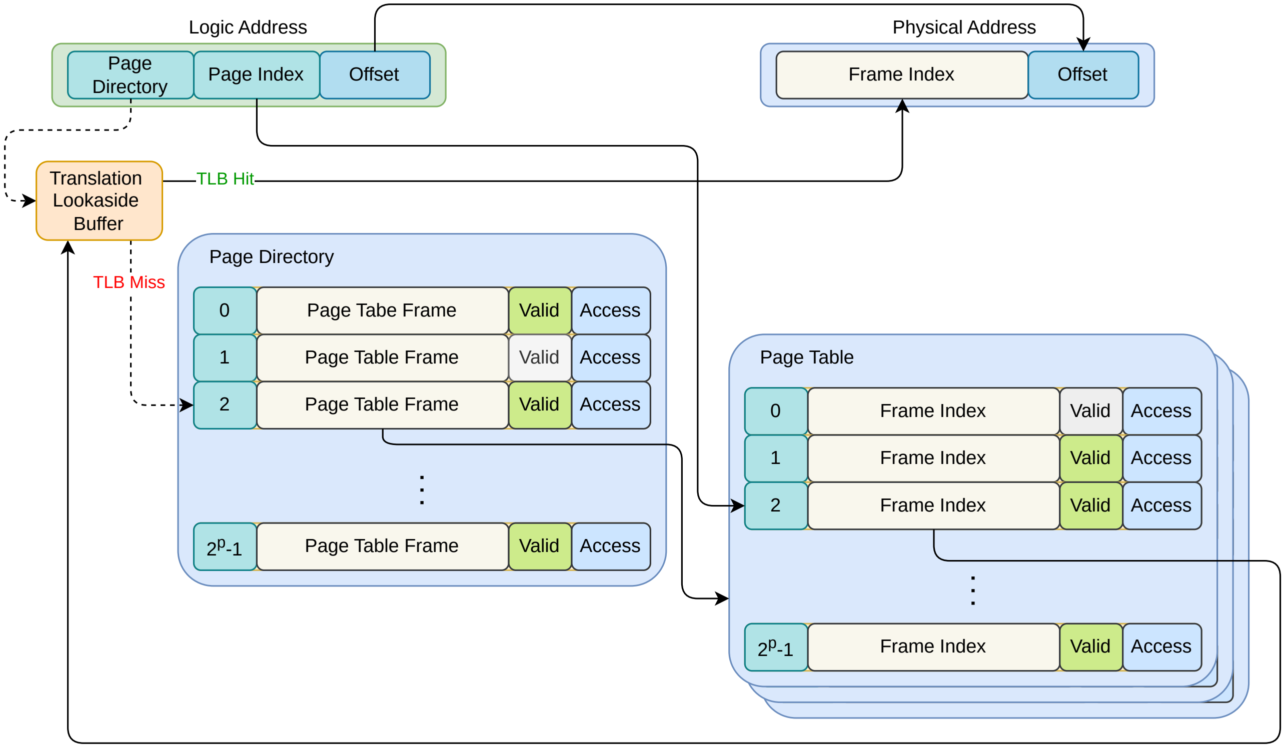
$$size_{table} = \frac{size_{ram}}{size_{page}}$$

- each table entry is 4B
- the address space is 4GB (for 32 bits processors)

$$size_{table\_32\_bits} = \frac{2^{32}}{4 \times 2^{10}}$$

$$size_{table\_32\_bits} = 4MB$$

RAM was counted in MB when paging started being used



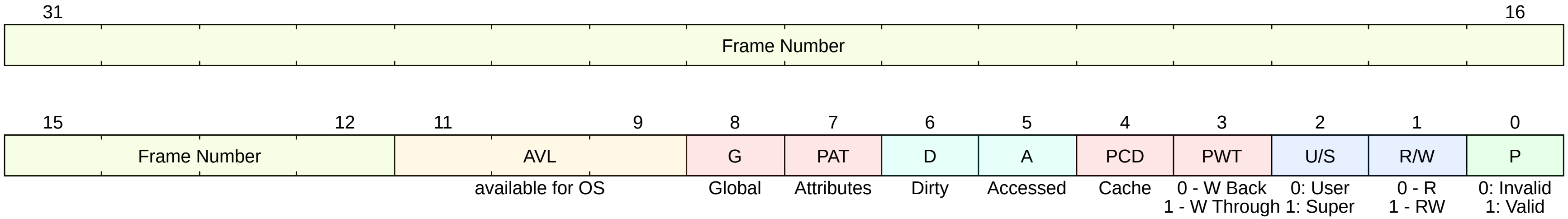
two levels, page directory and table, usually used for 32 bits systems

Page Table Entry

for x86 - 32 bits

this is one entry of the page table

- **P** - is the page's frame present in RAM?
- **R/W** - read only or read write access
- **U/S** - can the page be accessed in user mode?
- **D** and **A** - has this page been written since the OS has reset these bits?
- **AVL** - bits available for the OS to use, ignored by MMU



31	16
Frame Number	

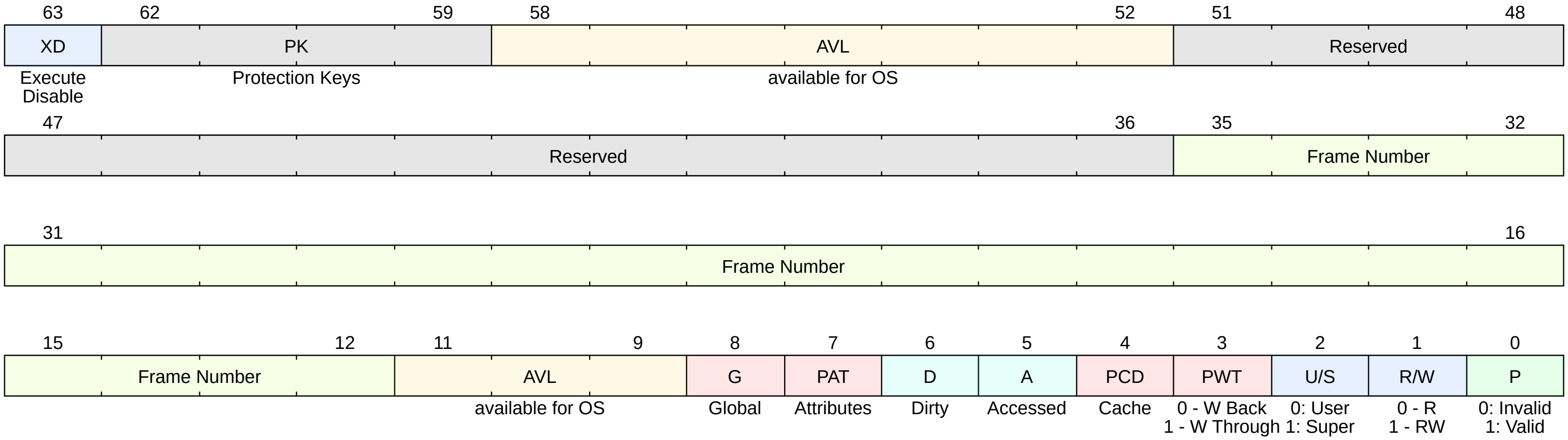
15	12	11	9	8	7	6	5	4	3	2	1	0
Frame Number		AVL		G	PAT	D	A	PCD	PWT	U/S	R/W	P
		available for OS		Global	Attributes	Dirty	Accessed	Cache	0 - W Back 1 - W Through	0: User 1: Super	0 - R 1 - RW	0: Invalid 1: Valid

Page Table Entry

for x86 - 32 bits with PAE

this is one entry of the page table using Physical Address Extension (*PAE*)

- **XD** - eXecute Disable (aka *DEP*), if set triggers a fault if an instruction is read from the page
- **PK** - Protection Keys, allows user mode to set protection (64 bit only)



63	62 59	58 52	51 48
XD	PK	AVL	Reserved
Execute Disable	Protection Keys	available for OS	

47 36	35 32
Reserved	Frame Number

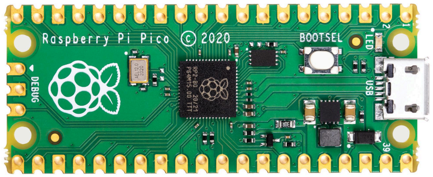
31 16
Frame Number

15 12	11 9	8	7	6	5	4	3	2	1	0
Frame Number	AVL	G	PAT	D	A	PCD	PWT	U/S	R/W	P
	available for OS	Global	Attributes	Dirty	Accessed	Cache	0 - W Back 1 - W Through	0: User 1: Super	0 - R 1 - RW	0: Invalid 1: Valid

Microcontroller (MCU)

Integrated in embedded systems for certain tasks

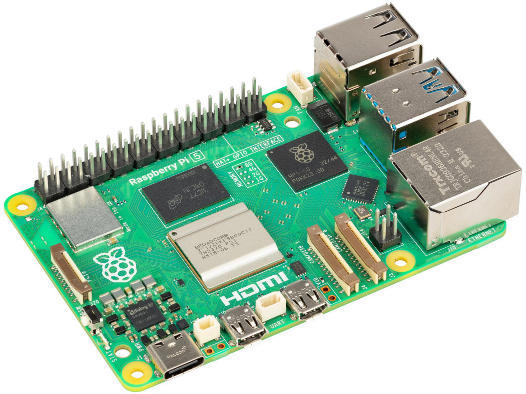
- low operating frequency (MHz)
- a lot of I/O ports
- controls hardware
- does not require an Operating System
- costs $0.1 - $25
- uses **Memory Protection Unit**



Microprocessor (CPU)

General purpose, for PC & workstations

- high operating frequency (GHz)
- limited number of I/O ports
- usually requires an Operating System
- costs $75 - $500
- uses **Memory Management Unit**



Conclusion

we talked about

- Memory Protection Unit
- Memory Management Unit