Memory Management
Exercise

Politecnico di Torino

Department of Control and Computer Engineering

System Programming - Sarah Azimi

CAD & Reliability Group
DAUIN– Politecnico di Torino

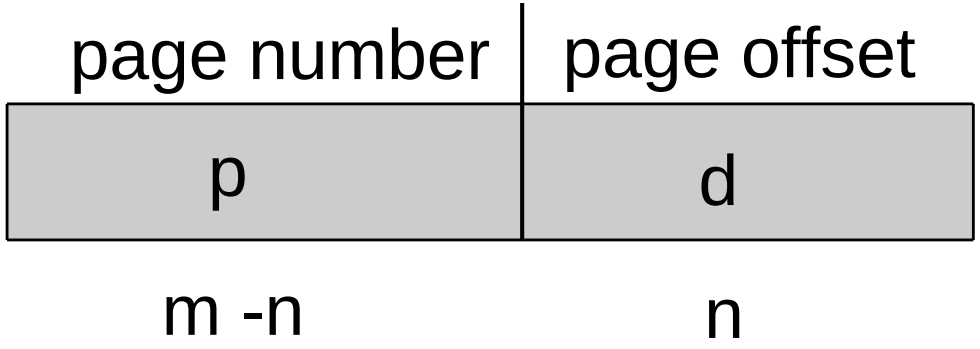
Exercise 1 (01/09/2018):

Sia dato un sistema di memoria virtuale con paginazione, nel quale vengono indirizzati i Byte. Il sistema dispone di **TLB** (Translation Look-aside Buffer), su cui si misura sperimentalmente un "**hit ratio" del 99 %.** La tabella delle pagine ("**page-table**") viene realizzata con uno schema **a due livelli**, nel quale un **indirizzo logico di 32 bit** viene suddiviso (da MSB a LSB) in 3 parti: **p1, p2, d, rispettivamente di 10 bit, 11 bit, 11 bit.** Non si utilizzano ulteriori strutture dati (quali tabelle di hash o inverted page table) per velocizzare gli accessi.

- Si dica che cosa si intende con "hit ratio"
- Si illustri lo schema della page-table e la sua dimensione complessiva, per un processo **P1** avente spazio di indirizzamento virtuale di **100 MByte**.
- Supponendo che la memoria RAM abbia tempo di accesso di 300 ns, si calcoli il tempo effettivo di accesso (EAT) per il caso proposto (hit ratio = 99 %)

Address Translation Scheme

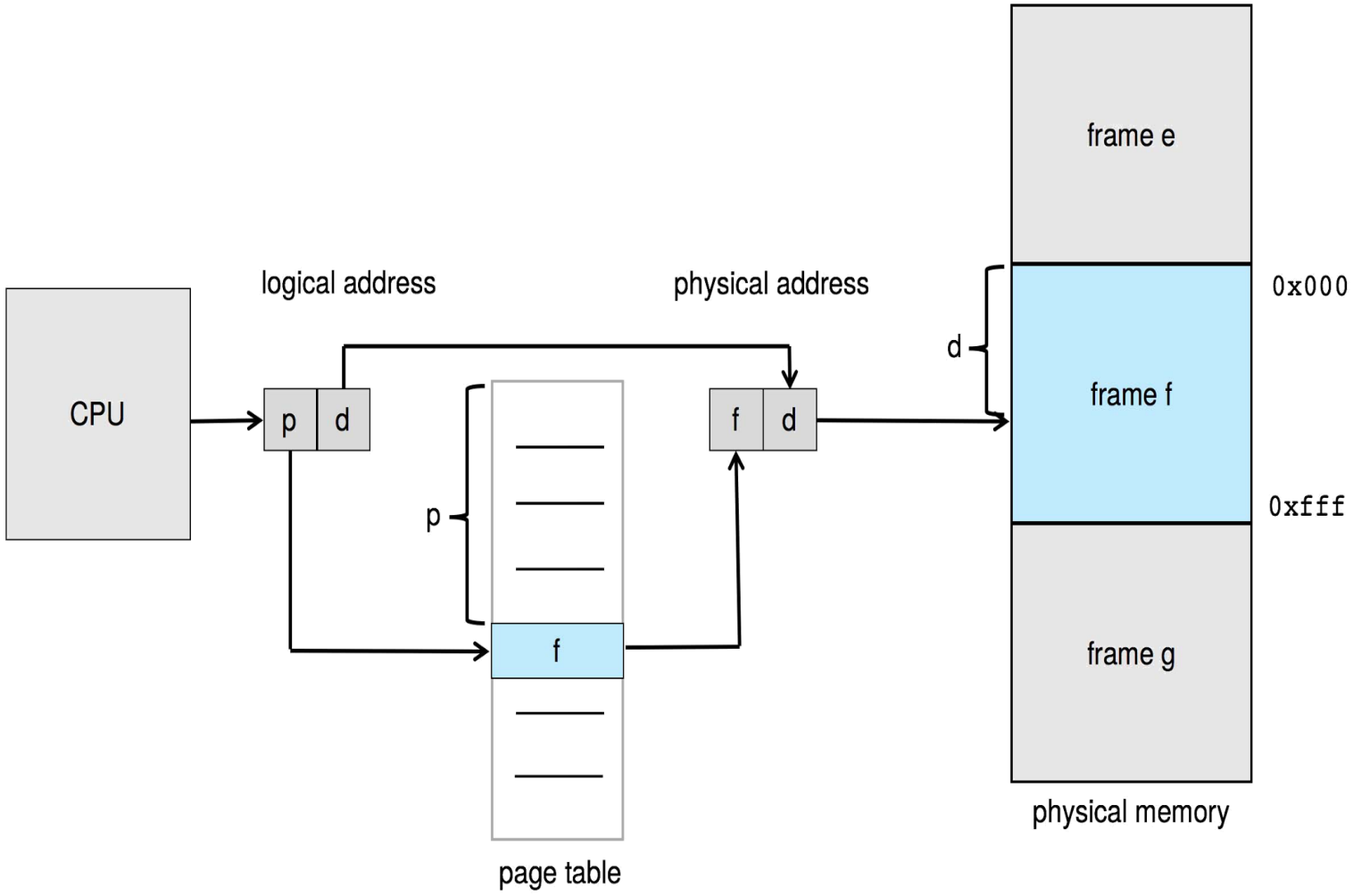
- Address generated by CPU is divided into:
  - Page number (*p*) – used as an index into a page table which contains base address of each page in physical memory
  - Page offset (*d*) – combined with base address to define the physical memory address that is sent to the memory unit



- For given logical address space 2^m and page size 2^n
- This information is used by Page Table

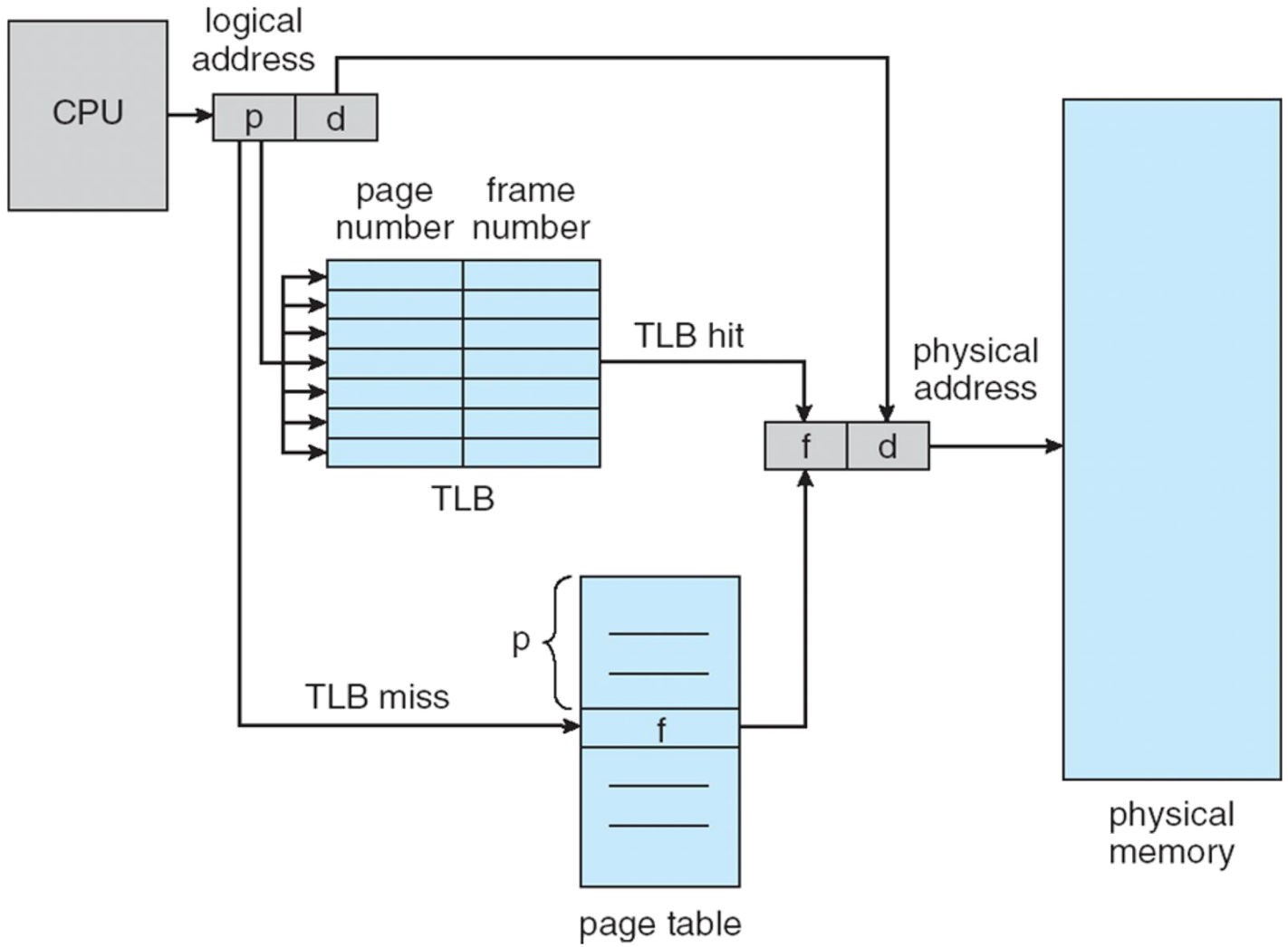
Paging Hardware

- The **page table** is a data structure used by the MMU to translate virtual memory addresses into physical memory addresses.
- The page table contains the base address of each frame in physical memory, and the offset is the location in the frame being references.
- The based address of a frame is combined with the page offset to define the physical memory address.



Paging Hardware With TLB

- The TLB contains only a few of the page-table entry.
- When a logical address is generated by the CPU, the MMU first checks if its page number is present in TLB.
- If the page number is found, its frame number is immediately available and used to access memory.
- These steps adds negligible performance penalty.
- If the page number is not in the TLB, **TLB miss**, address translation is done as before.



Effective Access Time

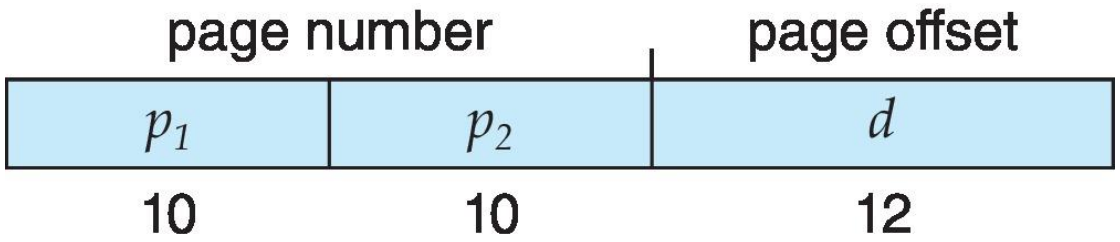
- The percentage of times that the page number of interest is found in the TLB is called the hit ratio.
  - An 80% hit ratio means that we find the desired page number in the TLB 80% of the time.
- Suppose that it takes 10 ns to access memory.
  - If we find the desired page in TLB then a mapped-memory access takes 10 ns
  - If we fail to find the page number in the TLB, then we must first access memory for the page table and frame number (10 ns) and then access to desired byte in memory (10 ns), for a total of 20 ns.
- Effective Access Time (EAT)

  $$EAT = 0.80 \times 10 + 0.20 \times 20 = 12 \text{ nanoseconds}$$

  - Suffering from a 20% slowdown in average memory access

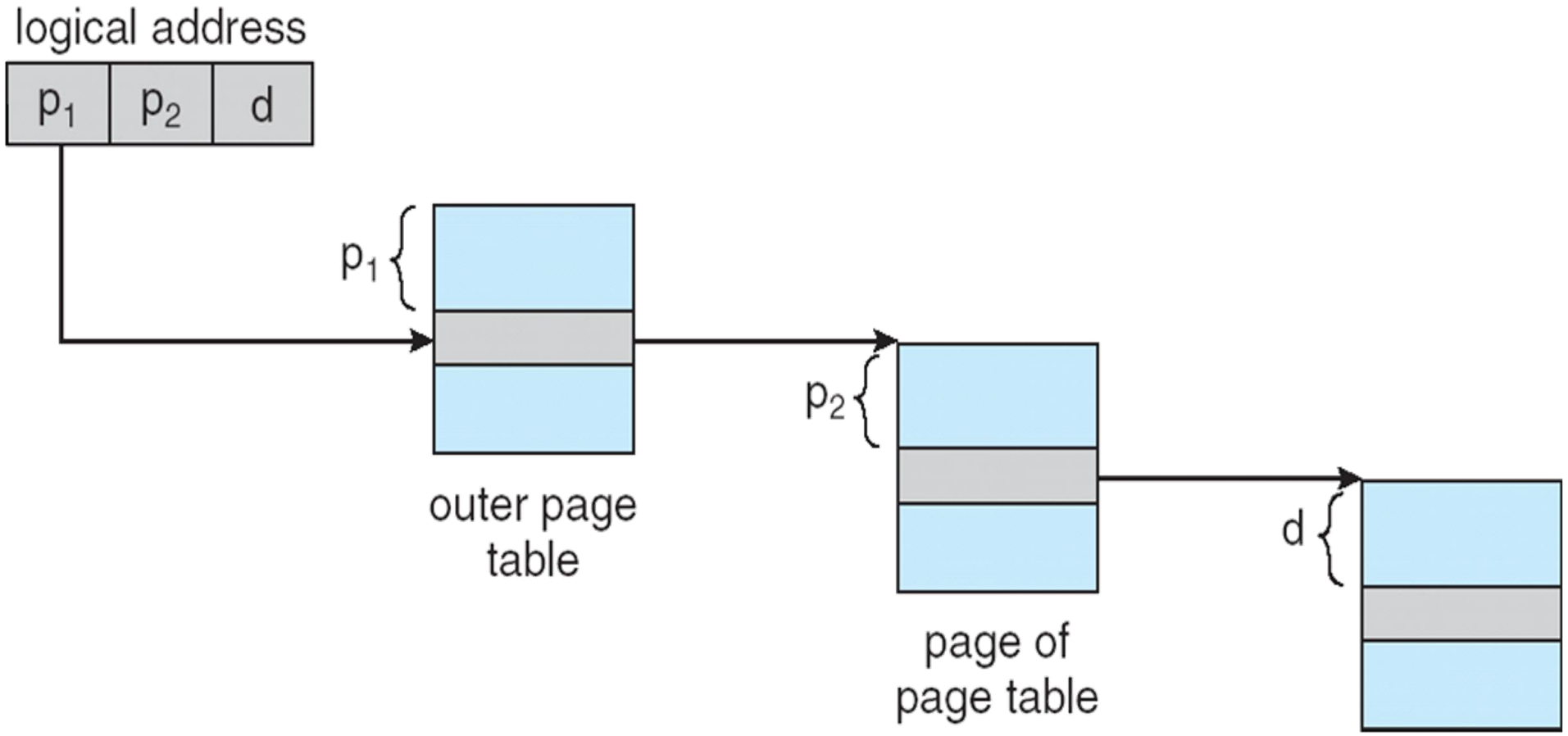
Two-Level Paging Example

- A logical address (on 32-bit machine with 4K page size) is divided into:
  - a page number consisting of 20 bits
  - a page offset consisting of 12 bits
- Since the page table is paged, the page number is further divided into:
  - a 10-bit page number
  - a 10-bit page offset
- Thus, a logical address is as follows:



- where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the inner page table
- Known as forward-mapped page table

Address-Translation Scheme



Exercise 1 (01/09/2018):

Sia dato un sistema di memoria virtuale con paginazione, nel quale vengono indirizzati i Byte. Il sistema dispone di **TLB** (Translation Look-aside Buffer), su cui si misura sperimentalmente un "**hit ratio" del 99 %.** La tabella delle pagine ("**page-table**") viene realizzata con uno schema **a due livelli**, nel quale un **indirizzo logico di 32 bit** viene suddiviso (da MSB a LSB) in 3 parti: **p1, p2, d, rispettivamente di 10 bit, 11 bit, 11 bit.** Non si utilizzano ulteriori strutture dati (quali tabelle di hash o inverted page table) per velocizzare gli accessi.

- Si illustri lo schema della page-table e la sua dimensione complessiva, per un processo **P1** avente spazio di indirizzamento virtuale di **100 MByte**.
- Si tenga conto che 100 MByte < 128 MByte = 2^{27} Byte e che pagine e frame hanno dimensione 2Kbyte = 2^{11} Byte (ricavate da d: 11 bit). Ipotizzando che ogni entry nelle tabelle delle pagine utilizzi 32 bit (4 Byte), una tabella delle pagine di secondo livello avrà 2^{11} caselle (11 da p2) e dimensione 2^{11}*4 Byte, mentre una tabella delle pagine di primo livello avrà 25 caselle (servono solo 5 bit su 10 di p1) e dimensione di 25*4 Byte.

Exercise 1 (01/09/2018):

Sia dato un sistema di memoria virtuale con paginazione, nel quale vengono indirizzati i Byte. Il sistema dispone di **TLB** (Translation Look-aside Buffer), su cui si misura sperimentalmente un "**hit ratio" del 99 %.** La tabella delle pagine ("**page-table**") viene realizzata con uno schema **a due livelli**, nel quale un **indirizzo logico di 32 bit** viene suddiviso (da MSB a LSB) in 3 parti: **p1, p2, d, rispettivamente di 10 bit, 11 bit, 11 bit.** Non si utilizzano ulteriori strutture dati (quali tabelle di hash o inverted page table) per velocizzare gli accessi.

- Supponendo che la memoria RAM abbia tempo di accesso di 300 ns, si calcoli il tempo effettivo di accesso (EAT) per il caso proposto (hit ratio = 99 %)
- EAT = (0.99 * 300 + 0.01 * 3 * 300) ns = 1.02 * 300ns = 306 ns
- ATTENZIONE: nel caso di TLB MISS ci sono 3 accessi non 2 (la page table e a due livelli, dunque 2 accessi per la PT più uno per l'accesso voluto in RAM)

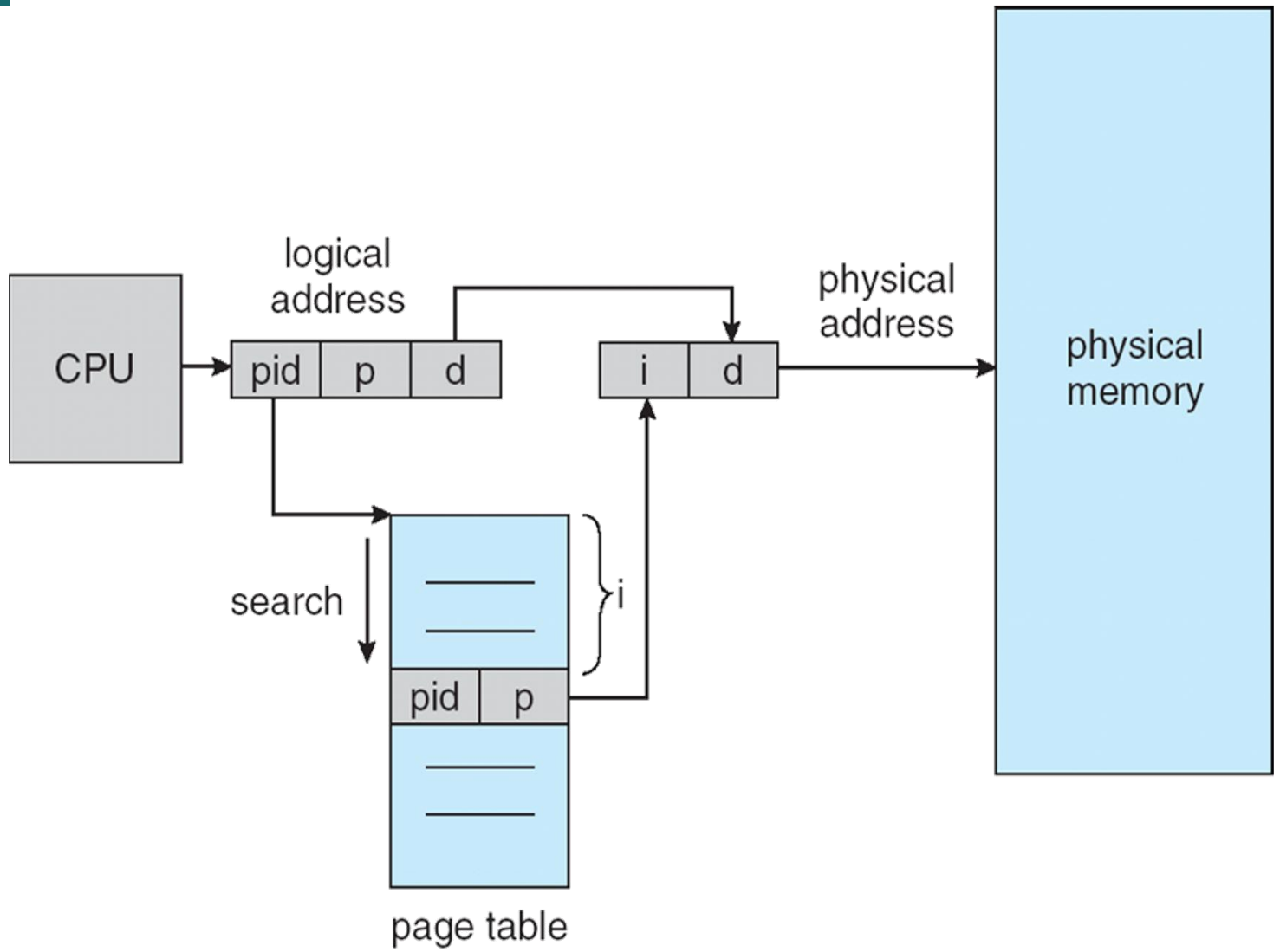
Exercise 2 (25/06/2018):

Sia dato un processo avente **spazio di indirizzamento virtuale di 32 GB**, dotato di **8GB di RAM**, su una architettura a 64 bit (in cui si indirizza il Byte), con gestione della memoria paginata (**pagine/frame da 1KB**). Si vogliono confrontare una soluzione basata su **tabella delle pagine standard** (una tabella per ogni processo) e una basata su **IPT**. Si calcolino le dimensioni della tabella delle pagine (ad un solo livello) per il processo e della IPT. Si ipotizzi che il pid di un processo possa essere rappresentato su 16 bit. Si utilizzino 32 bit per gli indici di pagina e/o di frame. Si dica infine, utilizzando la IPT proposta (32 bit per un indice di pagina/frame).

Inverted Page Table

- Rather than each process having a page table and keeping track of all possible logical pages, track all physical pages
- One entry for each real page of memory
- Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs
- Use hash table to limit the search to one — or at most a few — page-table entries
  - TLB can accelerate access
- But how to implement shared memory?
  - One mapping of a virtual address to the shared physical address

Inverted Page Table Architecture



Exercise 2 (25/06/2018):

Sia dato un processo avente **spazio di indirizzamento virtuale di 32 GB**, dotato di **8GB di RAM**, su una architettura a 64 bit (in cui si indirizza il Byte), con gestione della memoria paginata (**pagine/frame da 1KB**). Si vogliono confrontare una soluzione basata su **tabella delle pagine standard** (una tabella per ogni processo) e una basata su **IPT**. Si calcolino le dimensioni della tabella delle pagine (ad un solo livello) per il processo e della IPT. Si ipotizzi che il pid di un processo possa essere rappresentato su 16 bit. Si utilizzino 32 bit per gli indici di pagina e/o di frame. Si dica infine, utilizzando la IPT proposta (32 bit per un indice di pagina/frame).

Page Table standard:

N pagine = 32GB/1KB = 32M (corrisponde al numero di indici di frame nella page table)

|Page Table| = 32M*4B = 128MB

IPT

La tabella, unica per tutti i processi, ha dimensione fissa, in quanto contiene un indice di pagina per ogni frame in RAM (si consideri, per semplicità, di gestire tutta la RAM, compresa quella assegnata al sistema operativo). Ogni riga della IPT contiene un indice di pagina (4B) e il pid del processo (2B)

N frame = 8GB/1KB = 8M

|IPT| = 8M*(4B+2B) = 48MB

Exercise 3 (13/01/2023):

Sia dato un sistema di memoria virtuale con paginazione, nel quale vengono indirizzati i Byte. Il sistema dispone di TLB (Translation Look-aside Buffer), su cui si misura sperimentalmente un “hit ratio” del 90%. La tabella delle pagine (“page-table”) viene realizzata con uno schema a due livelli, nel quale un indirizzo logico di 64 bit viene suddiviso (da MSB a LSB) in 3 parti: p1, p2 e d, rispettivamente di 40 bit, 12 bit e 12 bit. Non si utilizzano ulteriori strutture dati (quali tabelle di hash o inverted page table) per velocizzare gli accessi. La memoria virtuale viene gestita con paginazione a richiesta.

Si risponda alle seguenti domande:

A) Supponendo che non sia presente una memoria cache, e che la memoria RAM abbia tempo di accesso di 200 ns, si calcoli il tempo effettivo di accesso (EAT) per il caso proposto (TLB hit ratio = 90%), assumendo che il tempo di accesso alla TLB sia trascurabile.

Exercise 3 (13/01/2023):

A) Supponendo che non sia presente una memoria cache, e che la memoria RAM abbia tempo di accesso di 200 ns, si calcoli il tempo effettivo di accesso (EAT) per il caso proposto (TLB hit ratio = 90%), assumendo che il tempo di accesso alla TLB sia trascurabile. (A) Calcolo EATPT con TLB hit ratio = 90%:)

TRAM = 200 ns (RAM access time)

hTLB = 0.9 (TLB hit ratio)

2 level (hierarchical) PT => 2 reads for PT lookup

EATPT = hTLB*TRAM + (1-hTLB)*3TRAM = (1 + 2*(1-hTLB))TRAM = 1.2 * TRAM = 240 ns

Thanks

Politecnico di Torino

Department of Control and Computer Engineering

Questions?

sarah.azimi@polito.it