Gestione della memoria - II

Michele Favalli

Cache multilivello

- Per ottenere un miglior compromesso fra probabilità di miss e tempi di accesso, si inseriscono ulteriori livelli di cache
- Livello 1: piccolo e veloce (e.g. 16-64 KB, 1 ciclo)
  - tipicamente si hanno due cache separate per codice e dati
- Livello 2: più grande e più lento (e.g. 256 KB – 2 MB, 2-6 cicli)
- Le CPU moderne hanno tipicamente almeno 3 livelli di cache

Memoria Virtuale

- Problema: con 32 bit di indirizzo lo spazio di indirizzamento sarebbe di 2^{30} parole
  - tale spazio era decisamente maggiore delle dimensioni delle DRAM di allora
  - però ai programmatori avrebbe fatto comodo
- La memoria virtuale da al programmatore l'illusion di avere una memoria più grande della DRAM utilizzata come memoria principale (o memoria fisica)
- ***In pratica, la memoria principale funziona come una cache rispetto a quella virtuale (di massa)***
- La memoria di massa è molto lenta, nel caso di hard-disk servono ms per accedere a un indirizzo
- I solid-state disk riducono notevolmente questi tempi con costi maggiori

Memoria virtuale

- **Indirizzi virtuali**
  - I programmi usano indirizzi virtuali che estendo quelli fisici della DRAM
  - Lo spazio degli indirizzi virtuali è memorizzato su disco
  - Una parte degli indirizzi virtuali è memorizzato nella DRAM
  - La CPU traduce gli indirizzi virtuali in indirizzi fisici della DRAM
  - I dati non presenti nella DRAM vengono recuperati dal disco
- **Protezione della memoria**
  - Ogni programma ha una sua mappatura da indirizzi virtuali a indirizzi fisici (viene decisa dal sistema operativo)
  - Due programmi possono utilizzare lo stesso indirizzo virtuale per dati diversi
  - I programmi non necessitano informazioni su altri programmi in esecuzione
  - Un programma non può alterare lo stato della memoria degli altri

Analogie fra cache e memoria virtuale

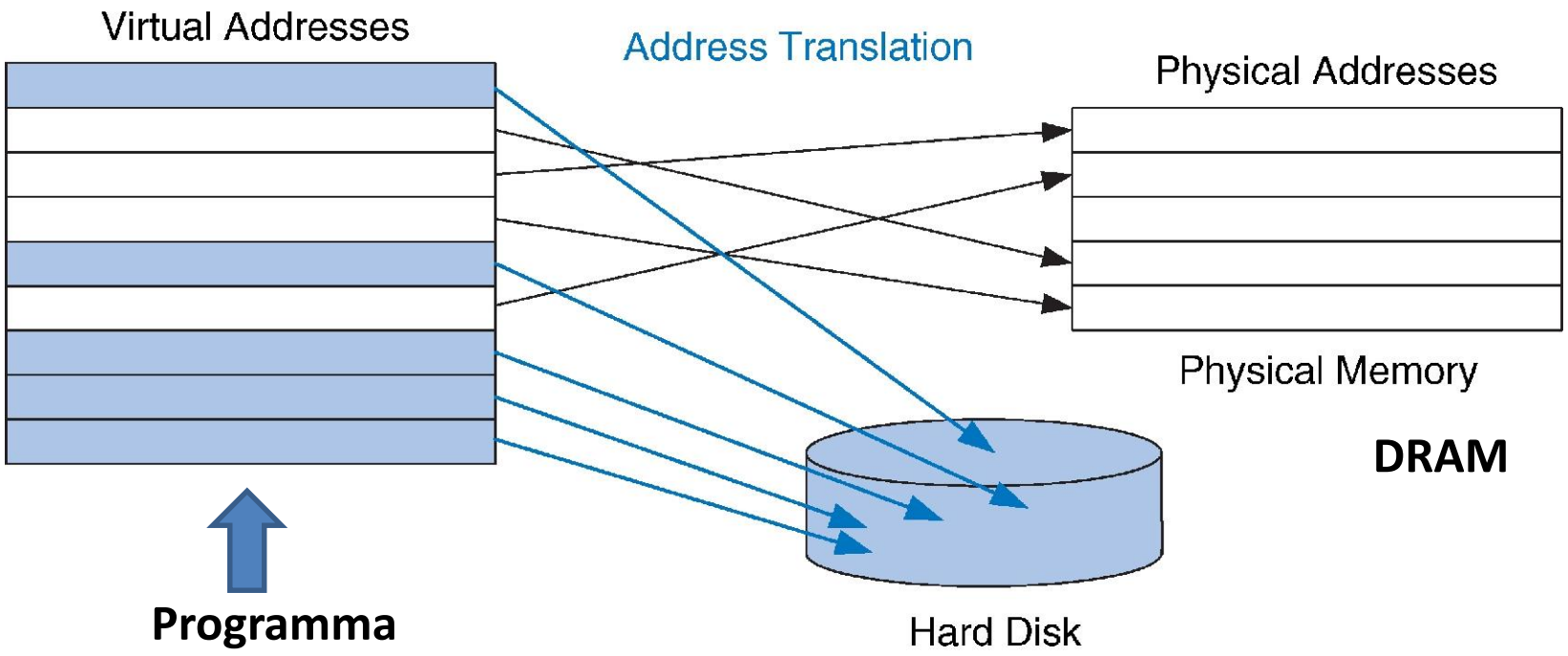
Cache	Virtual Memory
Block	Page
Block Size	Page Size
Block Offset	Page Offset
Miss	Page Fault
Tag	Virtual Page Number

La DRAM fa da cache alla memoria di massa

Definizioni

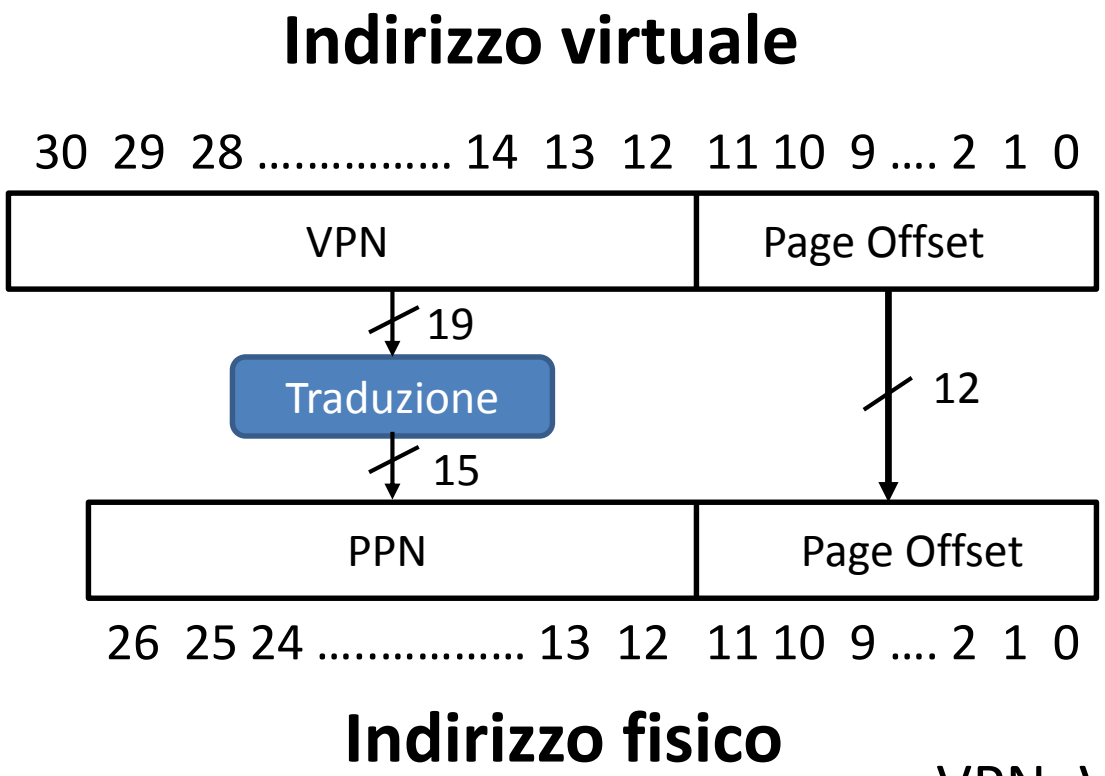
- **Page size:** quantità di memoria trasferita dalla memoria di massa alla DRAM per volta
  - si noti che le memorie di massa (HDD) impiegano molto tempo per trovare un indirizzo, ma poi sono relativamente veloci a estrarre blocchi di dati a partire da tale indirizzo
- **Address translation:** determina la locazione fisica di un indirizzo virtuale
- **Page table:** lookup table utilizzata per tradurre gli indirizzi virtuali a quelli fisici

Virtual & Physical Addresses



- La maggior parte degli accessi sono delle hit in memoria principale
- Comunque il programma può vedere uno spazio di indirizzamento maggiore della memoria principale

Traduzione degli indirizzi



VPN=Virtual Page Number

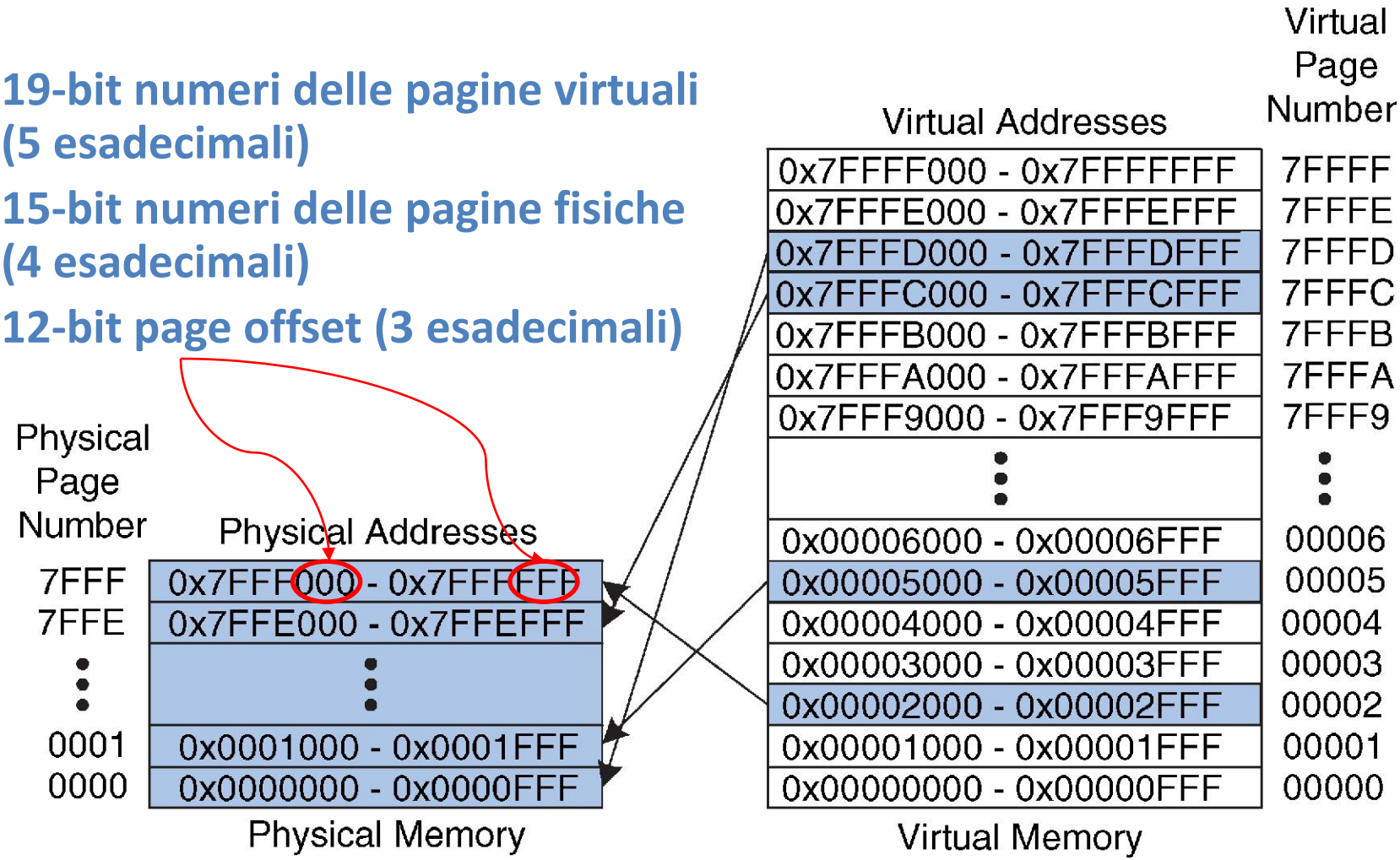
PPN=Physical Page Number

Esempio

- **Sistema:**
  - Dimensione della memoria virtuale: 2 GB = 2^{31} byte
  - Dimensione della memoria fisica: 128 MB = 2^{27} byte
  - Page size: 4 KB = 2^{12} byte
- **Organizzazione:**
  - Indirizzo virtuale: **31** bit
  - Indirizzo fisico: **27** bit
  - Page offset: **12** bit
  - # pagine virtuali = $2^{31}/2^{12}$ = $\mathbf{2^{19}}$ (VPN = 19 bits)
  - # pagine fisiche = $2^{27}/2^{12}$ = $\mathbf{2^{15}}$ (PPN = 15 bits)

Esempio

- **19-bit numeri delle pagine virtuali (5 esadecimali)**
- **15-bit numeri delle pagine fisiche (4 esadecimali)**
- **12-bit page offset (3 esadecimali)**

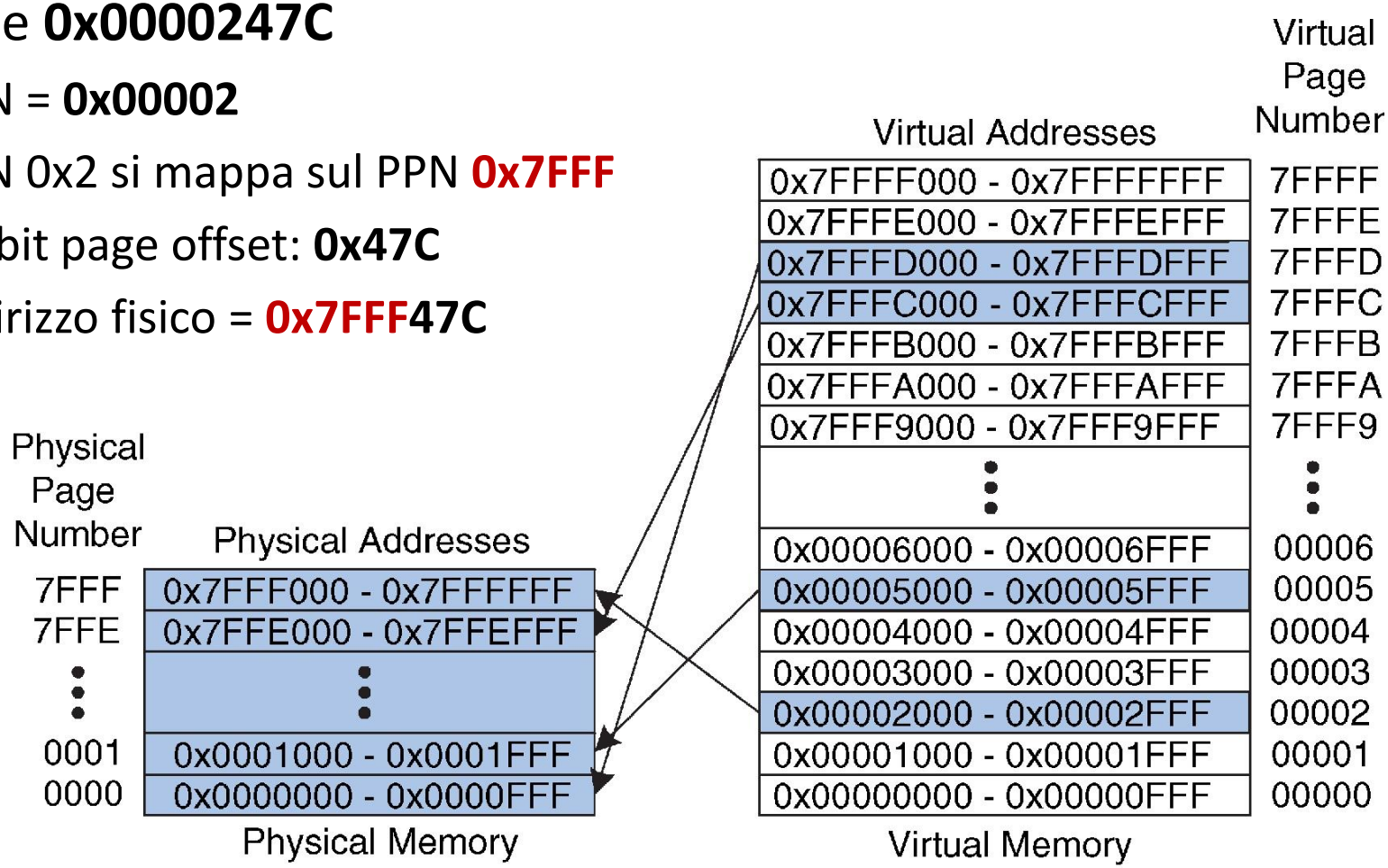


nota: gli indirizzi virtuali arrivano a 7 come massima cifra esadecimale più significativa perché sono costituiti da 31 bit

Virtual Memory Example

Indirizzo fisico dell'indirizzo virtuale **0x0000247C**

- VPN = **0x00002**
- VPN 0x2 si mappa sul PPN **0x7FFF**
- 12-bit page offset: **0x47C**
- Indirizzo fisico = **0x7FFF47C**



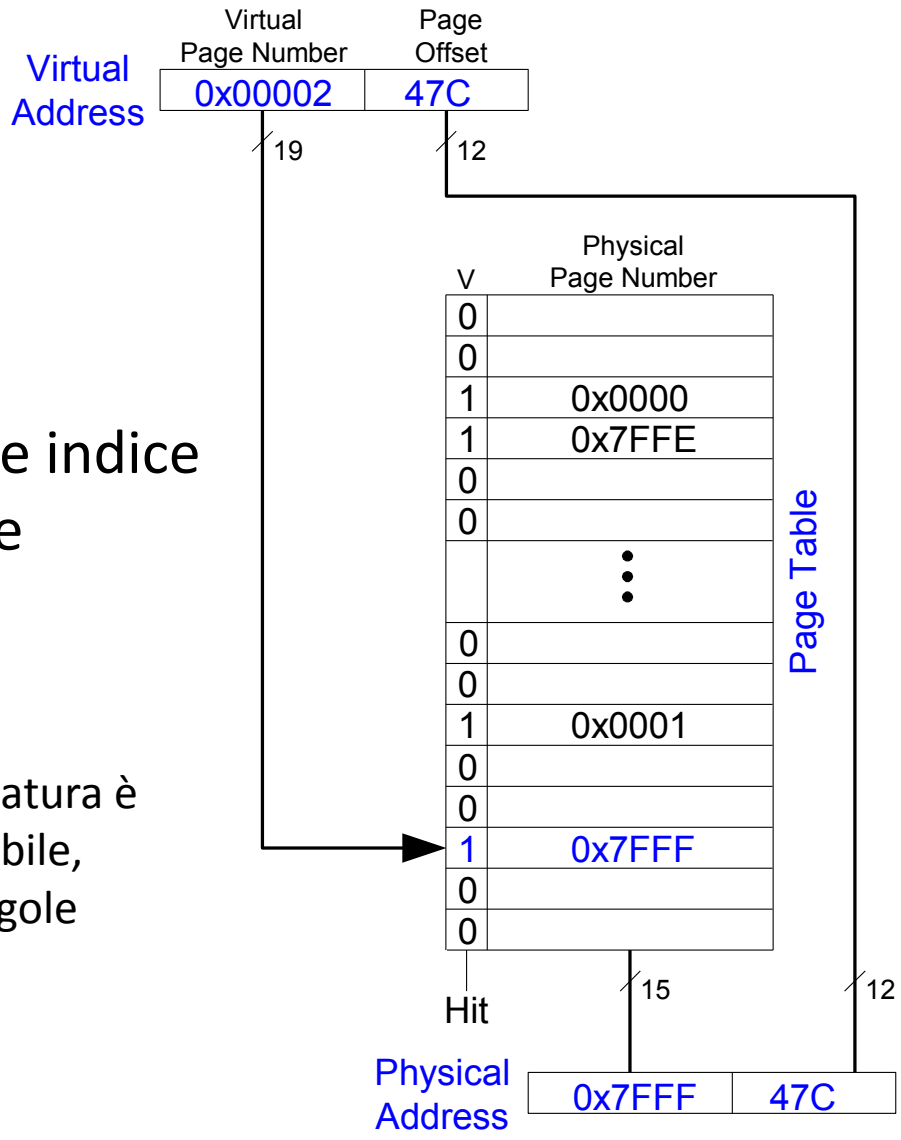
Realizzazione della traduzione

- **Page table**
  - Un elemento per ciascuna pagina virtuale
  - Campi:
    - **Valid bit:** 1 se la pagina è in memoria fisica
    - **Physical page number:** dove si trova la pagina in memoria fisica

Esempio

Il VPN si usa come indice nella lookup table

Si ricorda che la mappatura è solo un esempio possibile, qui non vediamo le regole per realizzarla

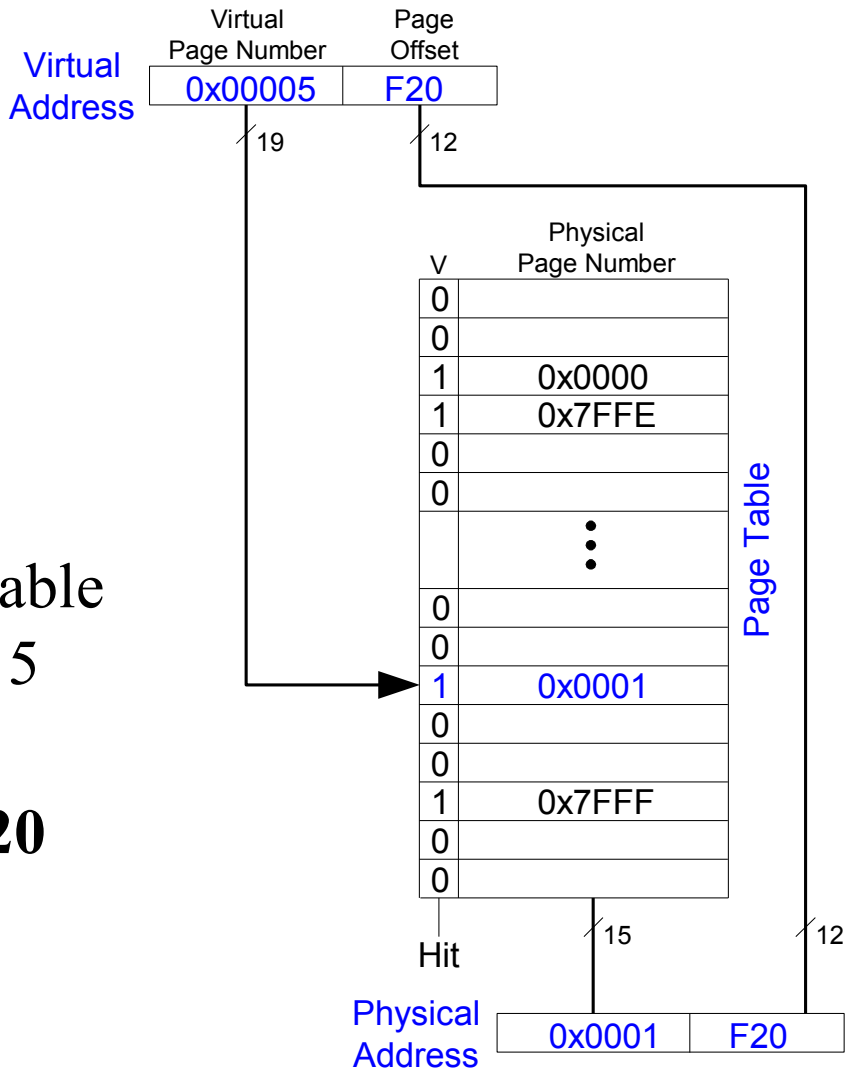


Esempio

Trovare l'indirizzo fisico correspondente a

0x00005F20

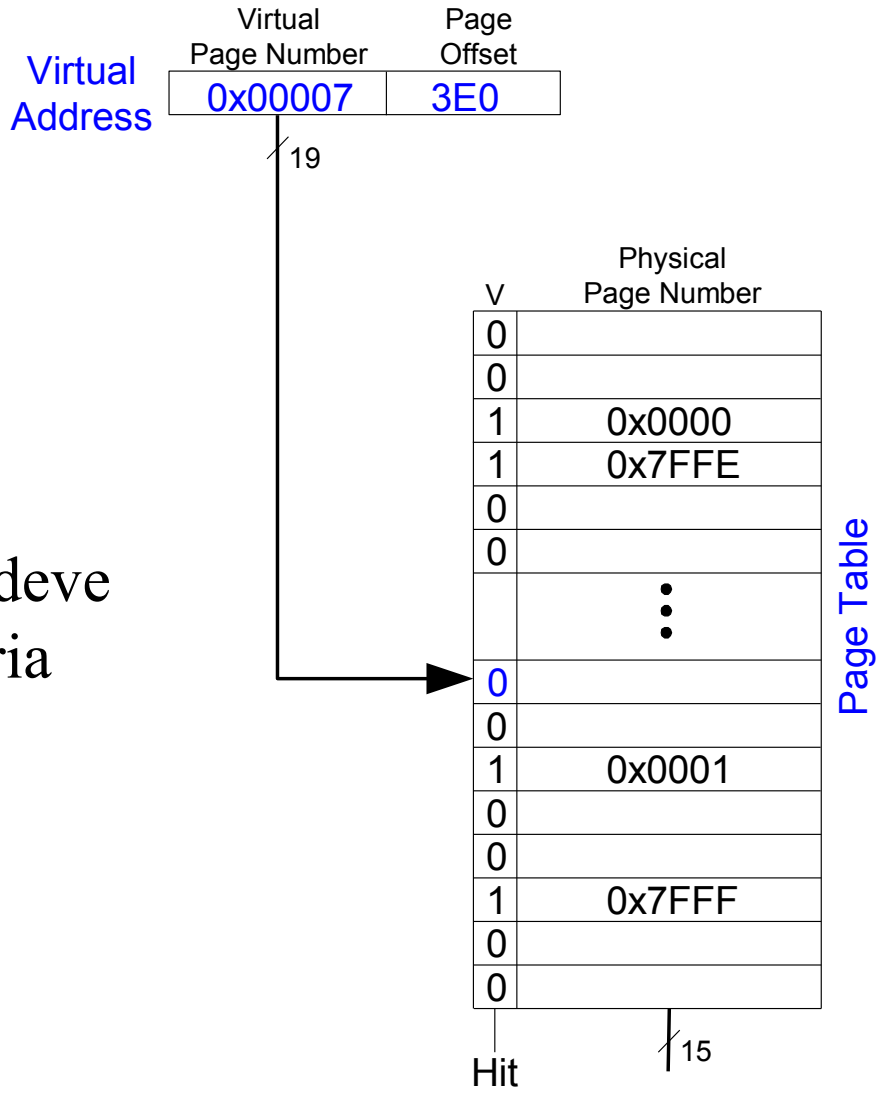
- VPN = **5**
- La entry 5 nella page table corrispondente a VPN 5 => pagina fisica **1**
- Indirizzo fisico: **0x1F20**



Un secondo esempio

Trovare l'indirizzo fisico corrispondente a **0x73E0**

- VPN = **7**
- La entry 7 non è valida
- La pagina virtuale page deve essere caricata in memoria dal disco



Compromessi di progetto nella page table

- **La page table ha dimensioni non trascurabili**
  - tipicamente si trova nella memoria fisica
- Una load/store richiede 2 accessi alla memoria principale
  - uno per la traduzione (page table)
  - uno per accedere al dato (dopo la traduzione)
- Si dimezzano le prestazioni della memoria
  - ***serve una soluzione …***

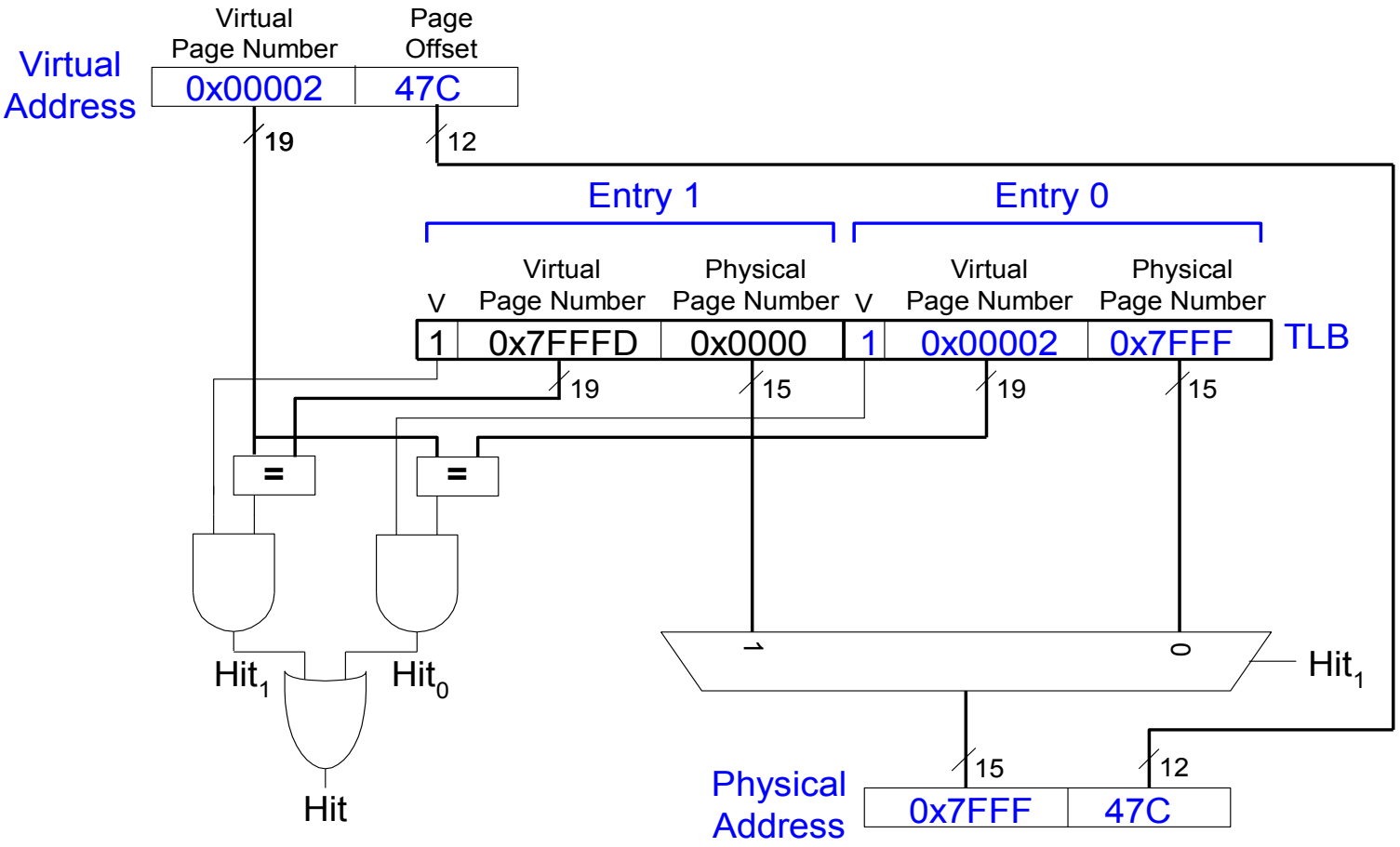
Translation Lookaside Buffer (TLB)

- Piccola cache contenente le traduzioni più recenti
- Nel caso ideale riduce il numero di accessi in memoria richiesti da load e store da 2 a 1

TLB

- Anche nel caso degli accessi alle pagine c'è un elevata località temporale
  - la dimensione delle pagine è piuttosto grande in modo che accessi in memoria consecutivi risultino in accessi alla stessa pagina (località spaziale degli accessi in memoria principale)
- TLB
  - piccolo: accesso in meno di un ciclo di clock
  - Tipicamente da 16 a 512 elementi
  - Organizzato in maniera pienamente associativa
  - Hit rate > 99 %
  - Riduce il costo degli accessi in memoria da 2 a 1

Esempio di TLB con due elementi



Protezione

- In un sistema operativo si ha la concorrenza fra processi multipli (ciascuno corrisponde a un programma)
- Ciascun processo ha la sua page table
- Ciascun processo può accedere all'intero spazio di indirizzamento virtuale
- Un processo può solo accedere a pagine fisiche mappate nella sua page table

Mappatura dell'I/O in memoria

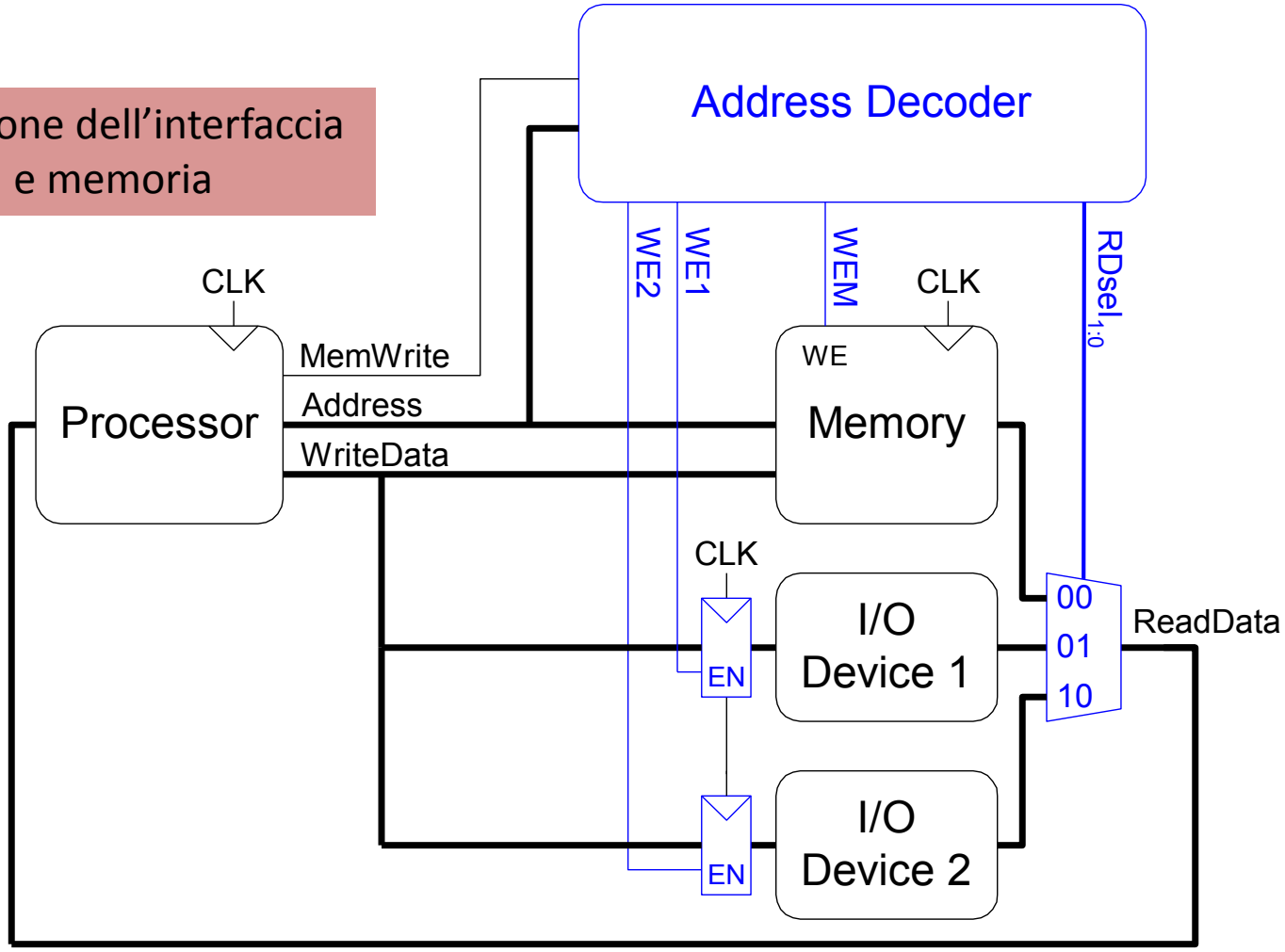
- Un calcolatore non è fatto solo di CPU e memoria (modello di Von Neummann)
  - come accedere all'I/O?
- Il processore accede ai dispositivi di I/O (tastiere, monitor, stampanti) nello stesso modo in cui accede alla memoria
- Ogni dispositivo di I/O viene assegnato a un insieme di indirizzi di memoria
- Quando in un accesso in memoria, si rileva tale indirizzo, I dati vengono letti o scritti su tale dispositivo invece che in memoria
- Una porzione dello spazio di indirizzamento è dedicato all'I/O

Mappatura dell'I/O in memoria

- **Address Decoder:**
  - guarda agli indirizzi e decide quale dispositivo o memoria comunica con il microprocessore
- **Registri di I/O:**
  - Contengono i dati scritti nei dispositive di I/O (in modo simile a una parola di memoria)
- **ReadData Multiplexer:**
  - Seleziona fra la memoria e I dispositive di I/O durante le operazioni di lettura

Memory-Mapped I/O

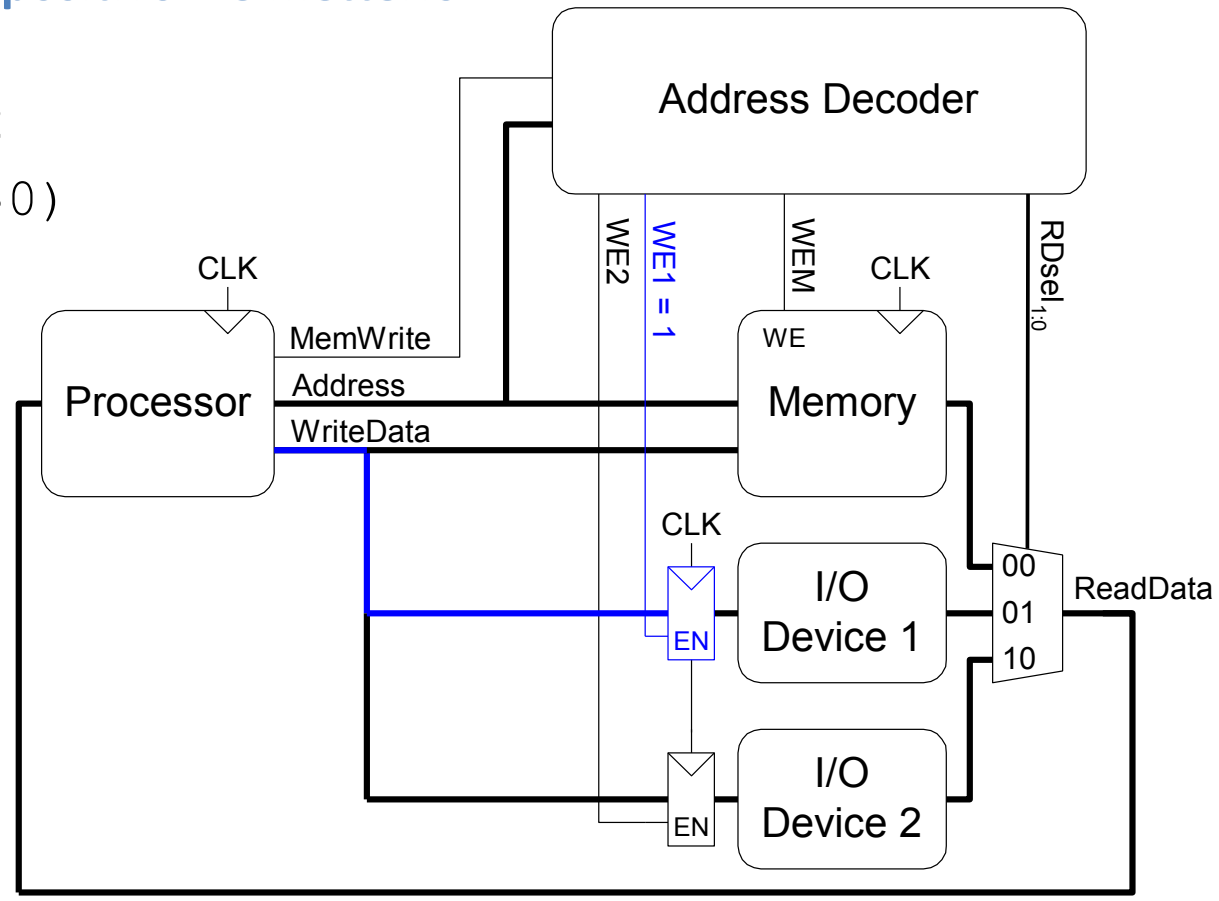
Estensione dell'interfaccia fra CPU e memoria



Memory-Mapped I/O

- Ipotesi: il dispositivo di I/O ha l'indirizzo 0xFFF4
  - **scrivere 42 nel dispositivo 1 ….**
  - **leggere il valore dal dispositivo 1 e metterlo in $t3**

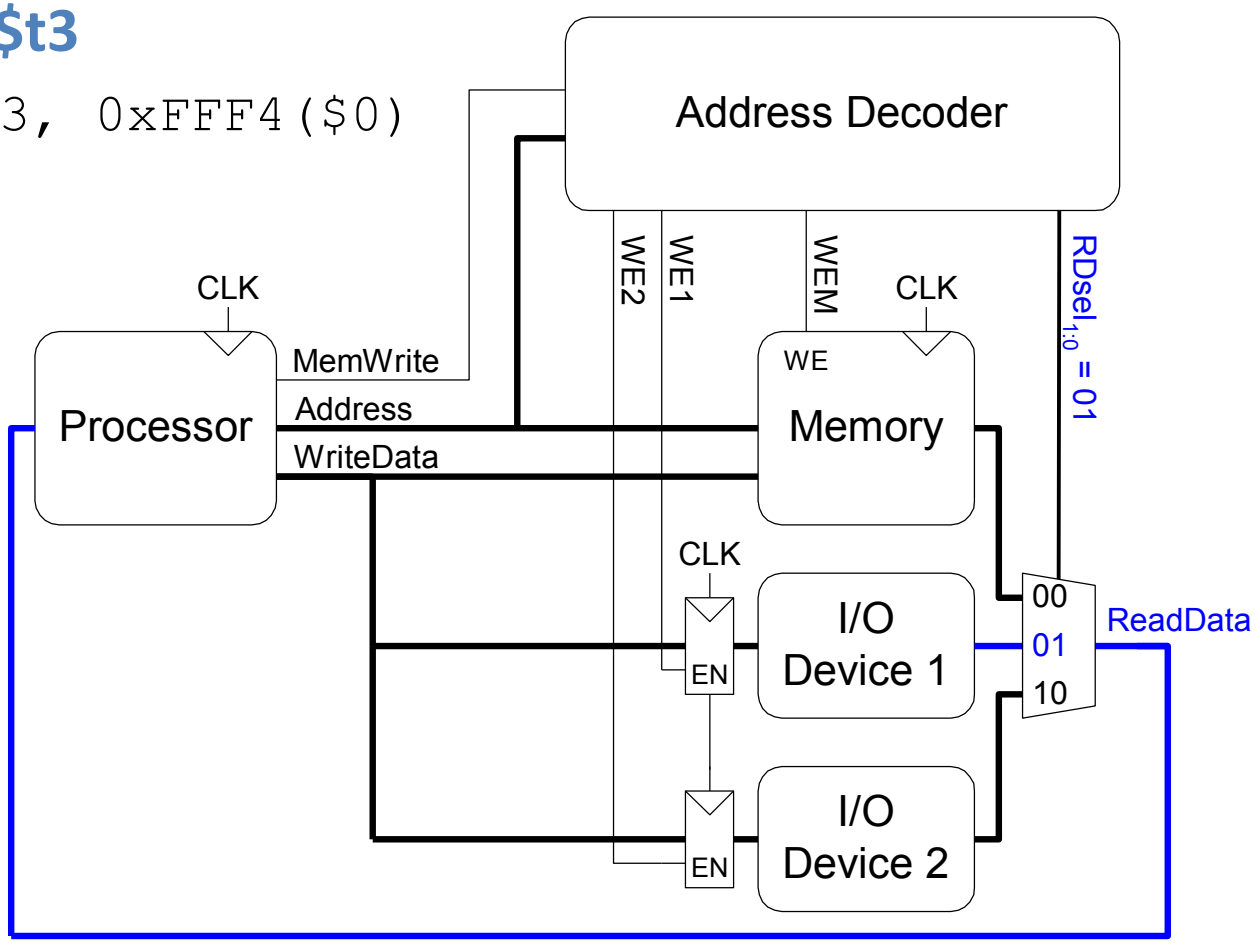
```
addi $t0, $0, 42
sw $t0, 0xFFF4($0)
….
```



Memory-Mapped I/O

- **Dopo un po' si legge il valore dal dispositivo 1 e lo si mette in $t3**

```
lw $t3, 0xFFF4($0)
```



Interrupt

- Alcuni dispositivi di I/O, come la tastiera, hanno un interfaccia che richiede letture/scritture da parte della CPU secondo tempistiche esterne
- Come gestirle?
  - verificare ciclicamente che non abbiano bisogno di comunicare => spreco di tempo
  - sistema di interruzioni
- Con gli interrupt un dispositivo di I/O comunica alla CPU la necessità di comunicare attraverso un opportuba logica (interrupt controller)
- Secondo determinate regole la CPU attiva un programma per comunicare con la periferica