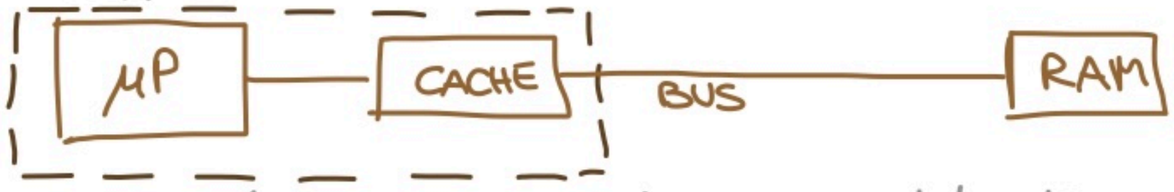
MEMORIA CACHE

↳ risolve problema realizzazione memorie grandi e veloci:
necessità di accedere a 1 parola in 1 ciclo di clock e di contenere quals. programma

→ si realizza una gerarchia di memoria a 2 livelli: A → mem. piccole e veloci
B → mem grandi ma lente

funziona se esec in A è non troppo veloce

← IDEA: tengo programma in mem. B e ← sposto in mem A parte da eseguire



↳ se in mem A manca dato, viene trasferito blocco da mem B

↳ concetto di LOCALITÀ: comportamento dell'accesso a mem dei programmi;

- TEMPORALE → un programma tende ad accedere agli stessi indirizzi di mem a cui ha acceduto recentemente;
- SPAZIALE → un prog. tende ad accedere con + prob. a indr. vicini a quelli già acceduti;

CARICO PROG IN B → QUANDO NEC. ACCESSO A 1 WORD CARICO BLOCCO IN A → GRZ A LOC. SPAZ. ACCEDO SPESSO A QUESTO BLOCCO → accessi HIT

→ VIENE RICHIESTO INDIRIZ. MANCANTE IN A → accesso MISS → VIENE CARICATO BLOCCO NUOVO DA B AD A

→ quando mem A è piena è nec. politica di sostituzione blocchi
- → casuale;
- → LRU, in rif. a princ spaziale;
- → FIFO, in rif a princ. temporale;

M=MISS penalty = t caric. nuovo blocco

→ l'idea base funziona se loc. dei progr. e dim. della mem. veloce permettono di raggiungere HIT rate suff. elev. da compensare MISS penalty;

→ indr in CACHE uguali a indr in mem. centrale:
↳ valid bit, indica se è una copia valida (presente) o libera

→ per MIPS considero 2 chache di primo livello:
- → ISTRUZIONI: se l'istr non è presente (istr, dati) viene sospesa esec → caricato blocco
  ↳ prelevato istr;
- → DATI: per lettura come istruzioni;
  per scrittura: se il dato non è presente in cache carico blocco e scrivo tram 2 modalità:
  - → write-through, non-differita quindi prima di canc. un dato lo scrivo in RAM
  - → write back, in differita scrivo su cache mentre su RAM solo al mom. della sost.

→ ORGANIZZAZIONE MEM. CACHE: (div. tipi di MAPPING, trad da mem. cent. a cache)
- indirizzam. diretto;
- complet. associativa;
- set-associativa;

1 → A INDIRIZZAMENTO DIRETTO

↳ ogni blocco della mem centrale è caricabile in cache.

- PROB: più blocchi possono necessitare di essere caricati in cache
  - o conflitto

→ **metodo mapping**: il blocco di indice (indr) j della mem centrale è caricabile solo nel blocco di indice (indr.): $\frac{\text{resto div. intera di } j}{\text{n° blocchi cache}}$

es. 2^3 blocchi cache: 3 bit indr

2^5 blocchi mem. cent: 5 bit indr

$\frac{2^5}{2^2} = 2^2$ blocchi di mem centrale mappati su cache

quindi in mem centrale ho 5 bit a indr:
- 2 bit mi dicono se è in cache
- 3 bit mi dicono indr cache

→ **etichetta**: N bit indr in mem centrale (2^N parole/byte)
K bit per ident. parola in RAM (2^K parole in blocco)
M bit per ident. blocco in cache (2^M blocchi in cache)

N-M-K	M	K
etichetta	blocco	parola

quale blocco in RAM è anche in cache

tag	index	offset

→ accesso semplice e veloce a blocchi

→ il blocco indirizzato viene attivato da un decoder che riceve in ing. l'indice del blocco in cache;

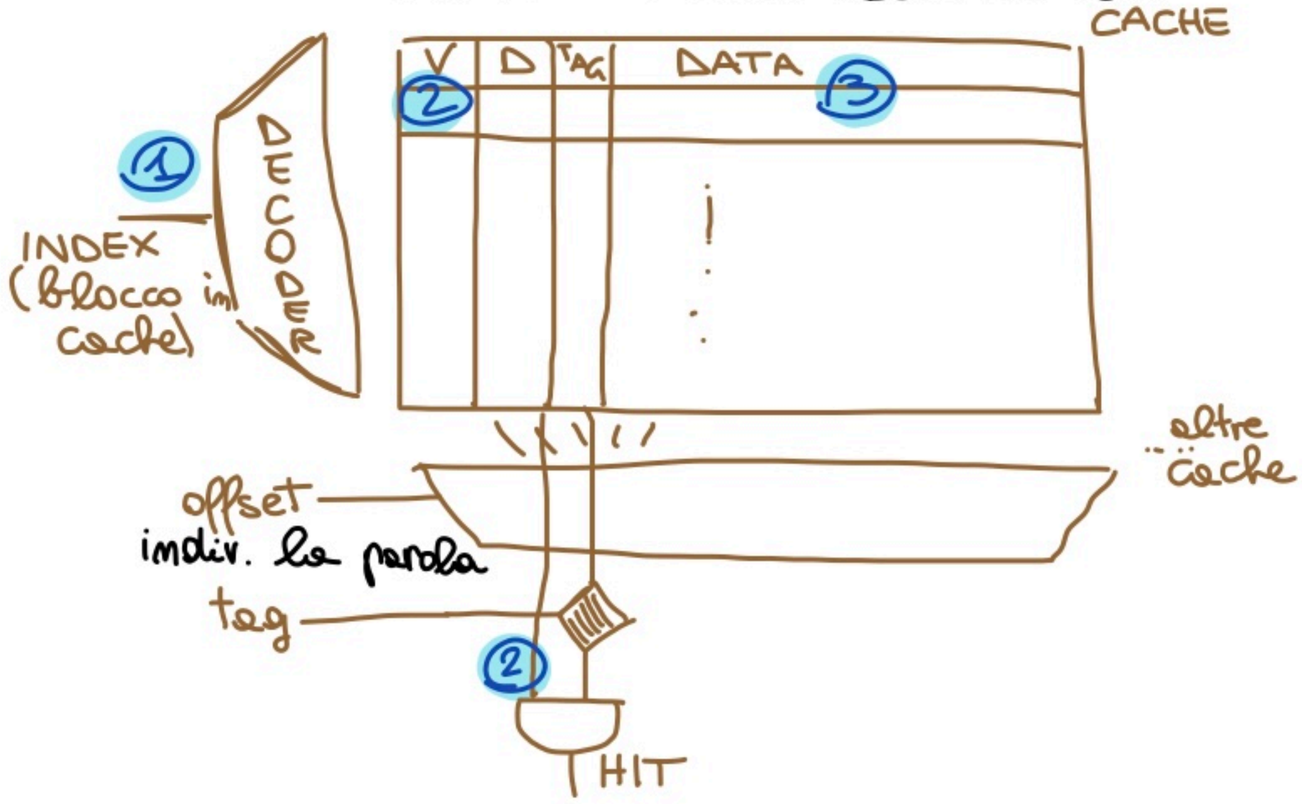
---

es. Mem centr. da 4 Gbyte $= 2^{32}$ → 32 bit a indr.
cache da 1 KB $= 2^{10}$ → 10 bit per blocco cache
blocchi da 32 Byte → 5 bit per parola

↳ $\frac{2^{32}}{2^5} = 2^{27}$ → 27 bit indice blocco

↳ $\frac{2^{10}}{2^5} = 2^5$ → 5 bit per ident. blocco cache

→ etichetta = 32 − 10 = 22 bit



V → valid bit, ovvero se ci sono dati validi

TAG → identifica l'indr. di mem corris ai dati memorizzati

D → dirty bit

DATA → copia dei dati

→ dato indr in mem centrale l'accesso a cache sarà:
- con index in decoder seleziono blocco cache; (1)
- con V capisco se è presente o meno il dato; (2)
- con tag accedo a parte DATA richiesta (3)

- indr mem a 32 bit
- parola da 4 Byte = 2 bit → OFFSET → ETIC = TAG = 32 − 2 − 10 = 20 bit
- cache da $2^{10} = 10$ bit → INDEX

2 → CACHE ASSOCIATIVA: un blocco può essere mem. ovunque nella cache, per cui tra indr in mem e blocco non c'è correlazione (NO MAPP.)

memoria con indr. N bit e blocchi cache da 2^M byte:

→ ETICHETTA = N − M

↳ per trovare etichetta corrispondente effettuo ricerca in parallelo tram mem assoc

X costose e non molti pro

→ è come una set-associativa a 1 linea

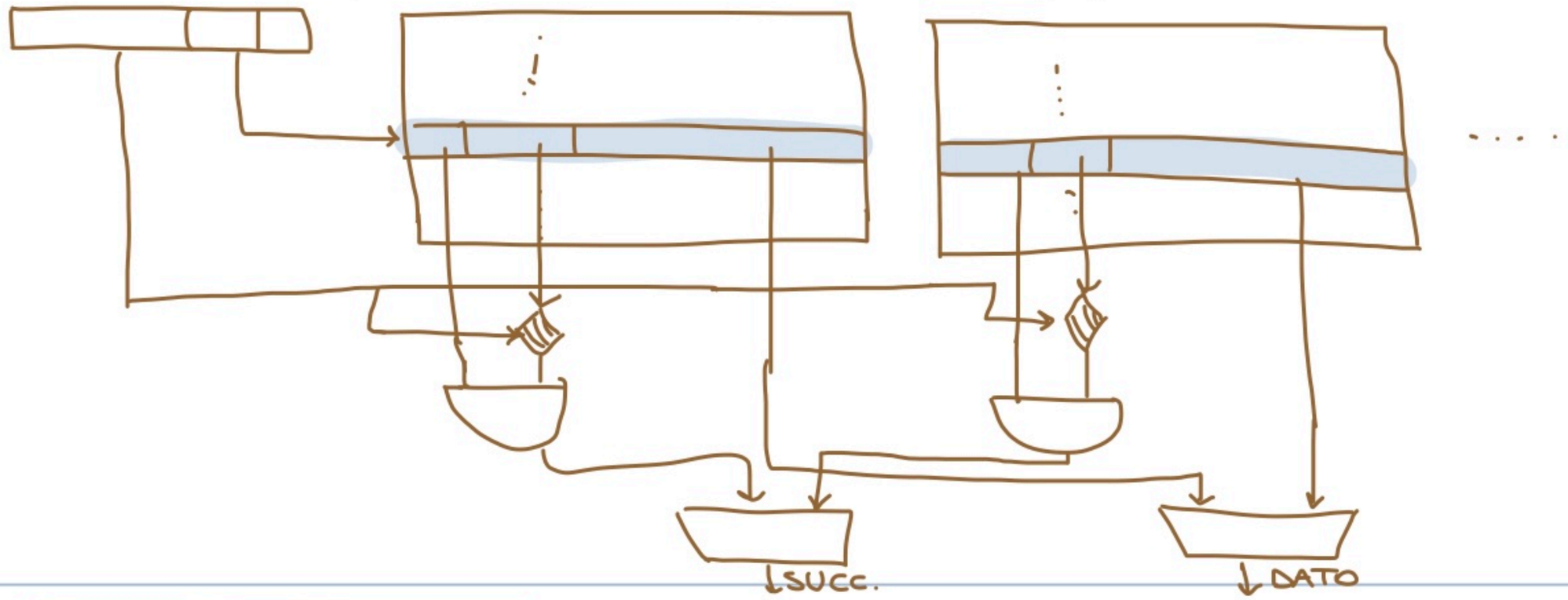
3 → SET-ASSOCIATIVO

↳ blocchi cache divisi in gruppi (set) da n (blocchi appunto)

↳ ogni blocco in mem può essere caricato IN UN SOLO GRUPPO PREFISSATO in un qualsiasi suo blocco (set-assoc. a n vie)

→ combina quindi indr. diretto + compl. assoc.

→ in questo caso INDEX identifica gruppo e da lì cerco tram tag;



→ TEMPO ACCESSO MEM: AMAT = tempo hit + freq miss * penalità

con burst: $T_{MISS} = T_{HIT} + T_{1° ISTR} + \left(\frac{\text{lung linea}}{\text{lung parola}} - 1\right) * T_{\text{sett. altre istr}}$

$T_{ACC,MED} = 1 * R_{HIT} + T_{MISS} * R_{MISS}$ $\qquad$ $T_{ACC.MED,TOT} = T_{IST} + T_{DATI}$