L-25 / 18-10-2023 /

Cache Memory

⊛ What, Why, How to use & Benefits?

⇒ Semiconductor memory.

⇒ Must faster than RAM (Main Memory)

⇒ Expensive, cost per bit is very high.

⇒ Because of that, capacity of cache is kept very small than RAM.

⊛ Why?

⇒ In a processing of an instruction, maximum time spend on Fetch and reading Data.

⇒ Time needs for fetch ⇒ depends on RAM and BUS.

✳ Let assume a program have 100 instruction.

All are register mode.

⇒ CPU can read only one instruction from RAM at a time.

Then, CPU needs to access RAM 100 times.

↳ Number of access = 100 times.

✳ Memory access time is 100 μsec.

⇒ then total access time = 100 × 100 μsec

∴ Average memory access time = $\frac{100 \times 100\ \mu sec}{100}$

(100 × 100 μsec: total access; single access time; 100: total instruction)

✳ If we don't have cache, then memory access time is the average memory access time. & depends on RAM.

↳ Average Access time = $\frac{\#\text{access} \times \text{Access time}}{\text{total instruction}}$

✱ Memory mode instruction:

OP R_x M R_2

inst. Data

This time ∴ CPU needs to access memory 100+100 = 200 times.

∴ Average Access time = $\frac{200 \times 100 \text{ nsec}}{100}$ = 200 μsec.

✱ OP R_x R_y R_z ⇒ 70.

OP R_x M R_z ⇒ 30.

RM inst. Data

then number of access time = 70 + 30 + 30

MM inst.

= 130 times.

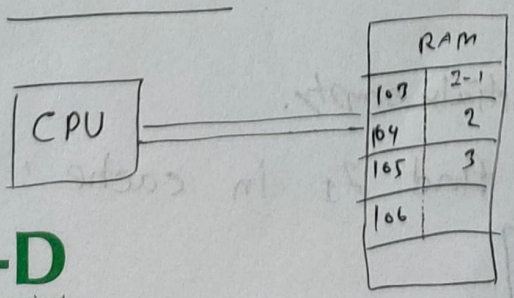
✱ OP R_x M_1 M_2 ⇒ 100 instruction

inst.

∴ number of memory access = 100 + 200 = 300 times.

Data Read

✱ Locality of Reference:



✳ Currently CPU is processing instruction-1, then, which instruction will be read next?

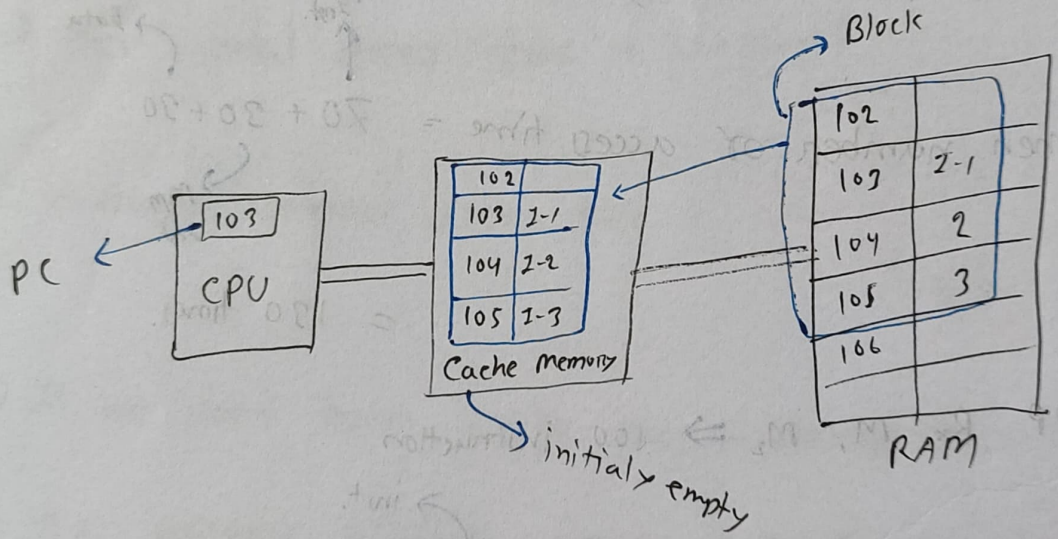
⇒ Then next memory address.

⇒ Instruction save next/ close to current one.

↳ Spatial locality reference

⇒ Recently executed instruction

↳ Temporal locality of reference.



✳ Steps:

i. Processor will search I_1 in cache by RAM address 103

ii. CPU does not find I_1 in cache ← initialy empty. : known as **cache miss.**

iii. In the event of cache miss, CPU will access RAM.

iv. A Block of RAM (including the instruction, CPU searching for) is transfer from RAM to Cache. → a section of RAM - capacity only few bytes. (4/2/16132)

v. CPU Reads I, from Cache memory.

* And the PC will updated by the address of next instruction.

* Whenever anything found in cache, known as ~~the~~ cache hit.

and in the event of cache hits CPU will simply read from cache memory.

* Block Transfer depends on locality of Reference.

✳ Lets assume that a program have 100 instruction.

80 instruction ⇒ Cache hit

Then, hit ratio = $\frac{80}{100} \times 100\% = 80\%$ } usually calculated in %.

= 0.8 (Normalized) }

✳ If Hit ratio is 80%.

then miss ratio is 20%.

= 0.2 (Normalized)

✳ Then, without cache,

number of memory access is 100 times.

And with cache,

number of memory access is 20 time ↳ cache miss

✳ Reduce the frequency of access to RAM.

L-26/30.10.2023/

Cache Memory

✳ Objective: Reduce the frequency of Access to Main Memory.

✳ I = 200

70% ⇒ Register Mode

30% ⇒ LOAD & STORE

∴ Number of access = 140 + 60 + 60

70% Read Ins.

30% Read Ins.

30% Read/Write memory

= 260 times.

✳ If cache memory used with Hit ratio 80%.

Then, $260 \times \frac{80}{100} = 208$ (read from only cache)

∴ number of access memory = 260 − 208

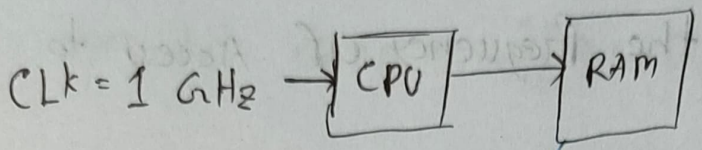
= 52

↳ Alternative way ⇒ $260 \times \frac{20}{100} = 52$

{ Hit Ratio ↑

number of access ↓

✱ For using cache, AVG memory acces time will be reduced.



50 ns ⇒ can be represent in clk cycle ⇒ 50 CLK cycle

∴ AVG access time = 50 ns (Without Cache)

⇒ if cache memory used with 80% hit ratio and access time (Hit Time) 5 ns.

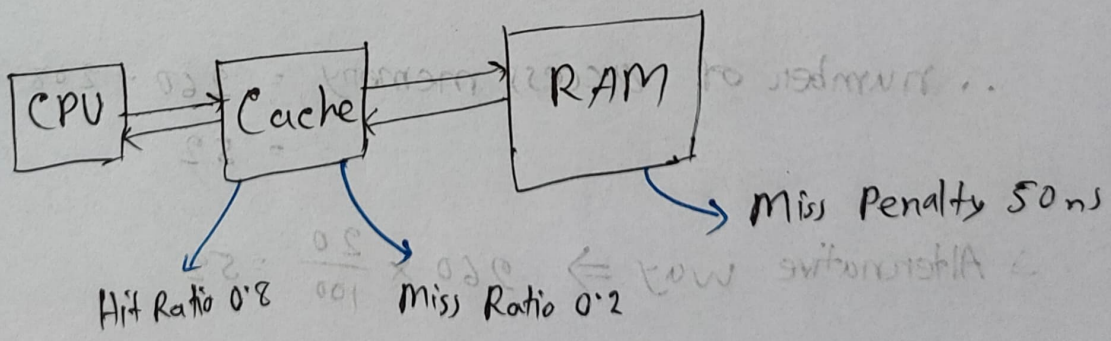
total inst. 1

In normalized form = $0.8 \times 5 + 0.2 \times 50 + 0.2 \times 5$

final read from cache

$= 15$ ns

✱



∴ AMAT = Hit Time + Miss Rate × Miss Penalty

= 5 ns + 0·2 × 50 ns

= 15 ns

In terms of CLK ⇒ AVG CPI or Effective CPI

✱ Memory stall cycles = Memory Access × miss penalty × miss Ratio

↳ total for a program

=

✱ Difference between, Memory stall cycles Vs AMAT

✱ Base CPI = Hit Time

✱ AVG CPI = Base CPI + AVG Instruction miss cycle + AVG data miss cycle

= 2 + 0·12 × 10 + 0·3 × 0·6 × 10

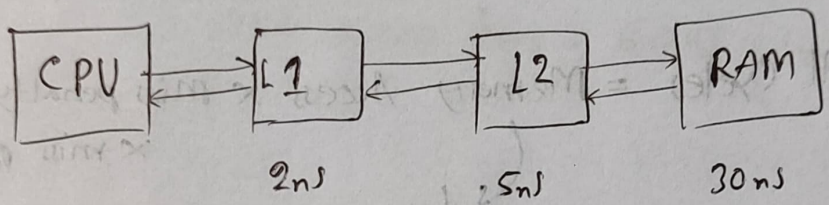
= 13·38

Multi level Cache!

⇒ CPU will always read from level 1.

Memory Size: L1 < 2 < 3 < RAM

Speed: L1 > 2 > 3 > RAM



AMAT = 0.8 × 2 + 0.2 × 0.9 × (2+5) + 0.02 × (30+5+2)

= 3.6

missing from Level 1 = 1 − 0.8
= 0.2

missing from level 2 = 0.2 − 02 × 0.9
= 0.02

AMAT = Hit Time$_{L1}$ + miss Rate$_{L1}$ × Miss Penalty

= Hit Time$_{L1}$ + Miss Rat$_{L1}$ [Hit time$_{L2}$ + miss rate$_{L2}$ × miss penalty$_{L2}$]

= 2 + 0.2 [5 + 0.1 × 30]

= 3.6

L-27/ 01.11.2023/

✳ From Slide,

CPI Stall = CPI base + L1 inst miss cycle + L1 data miss cycle

$= 2 + 1 \times 0.03 \times 100 + 0.4 \times 0.1 \times 100$

$= 9$ cycle

✳ Global Miss Rate

⇒ in case of multi level cache, the highest level cache miss rate is called Global miss rate.

✳ CPI Stall = CPI base + L1 inst miss cycle + L1 data miss cycle + L2 inst miss cycle + L2 data miss cycle

L-28/06.11.2023/

Cache Mapping

Three Type

(i) Direct

(ii) Fully Associative

(iii) Set- Associative

* When a block of RAM for instn. or Data is copied into cache, where it is copied in cache?

=> Transfer depends on first address of the block.

* Block : RAM divided into some equal section

RAM Size = 1 kB

Addressable location = 1024

Size of Block = 4 Bytes

Number of Block = $\frac{1024}{4} = 256 \ (0-255)$

* Blocks are identified by Block number.

✳ Starting address of block-0?

⇒ 0

✳ What are the addressable locations block-0 contains?

⇒ 0-3 (Decimal)

✳ Starting address of block-1?

⇒ 4

Contains: 4-7

✳ Starting Address = Block number × Block Size

= 1 × 4 = 4

✳ For block-15: 15 × 4 = 60 → Starting address

✳ Address contains:

j × Size of Block ⟶ j × Size of Block + Size - 1 ← Block

Starting address ⟶ Starting Address + Size - 1

* Given memory address, calculate Block number that address belongs?

⇒ Block Number = $\frac{\text{Memory Address}}{\text{Size of a Block}}$ (integer part only)

* Cache Block Size = RAM Block Size ⇒ Block Number
  - can be called as Line
  - Line number

* Cache Size = 64 Bytes
  Line/Block Size = 4 Bytes
  ∴ Total number of Line, m = 16 (index, 0-15)

* Each Block/Line of cache must be shared by many blocks of RAM.

* How does CPU identify which block of RAM is currently occupying a particular block/line of cache?

- There is a Tag field for each block/line in cache.
- The first address of block from RAM will be saved in Tag.
- Direct Maping formula:

Line, $i = j \text{ MOD } m$

m → total number of line

j → Block numbers of RAM

Example:

$j = 0 \Rightarrow i = 0$

$j = 16 \Rightarrow i = 0$

$j = 32 \Rightarrow i = 0$

} multiple of m are fixed for line 0.