29/10

Memory System

- For modern memory$\xrightarrow{\text{DRAM}}$, one memory access takes 150 - 200 cycles :((not 1 as before)
  - Not counting multiple memory accesses (just one, to fetch ic), each instruction will take ~200 cycles :(
  - Pipelining makes no sense, as a fetched instruction will take only #stages cycles more to complete.

- One possible solution :- 'Cache'

Between processor and memory, add a very small, but fast block of memory. This has, say, only 8 bytes (worth 2 instructions) and is made from more expensive SRAM.

SRAM memory is accessible in single cycle (must fetch consecutively written instructions)

- Fetch 8 bytes (2 instructions) from memory into SRAM (~200 cycles

Processor fetches	I_1	(200st cycle)	
	I_2	(201nd cycle)	---- pipelining
	I_3	(401rd cycle)	
	I_4	(402th cycle)	--- pipelining.

- Performance almost doubles by introducing SRAM, as at least 2 instructions are being pipelined.

eg Eg Program

```
for i in range(n):
    I1
    I2
    I3
    I4
```

Assume instruction do not access memory

If SRAM is 16 bytes, 4 instructions will be fetched in 200 cycles and stored in SRAM

For every subsequent iteration, we don't need to fetch again and instructions can be pipelined as before

CPI $\rightarrow$ 1 :)   as we are using only 4 instruction multiple times

Scanned by CamScanner

eg

```
for i in range (100):
    if i % 2 == 0 :-        I_1 I_2 I_3 I_4
    else                    I_5 I_6 I_7 I_8
```

- With 16 bytes of SRAM, we will have to fetch and replace the four instructions again and again.
  Total no. of cycles $= (200 + 1 + 1 + 1) \times 100$ ☹

- With 32 bytes of SRAM, we can get CPI $\rightarrow 1$, but only if all 8 instructions are consecutively written. Otherwise same as before.
  $(200 + 1 + 1 + 1 \cancel{+1+1+1+1}) \cancel{\times 50} + (1 + 1 + 1 + 1 \cancel{+1+1+1+1}) \times 49$

- With 2 SRAM chunks of 16 bytes each, we can get CPI $\rightarrow 1$
  The instructions in each individual chunk must be consecutive.
  $(200 + 1 + 1 + 1 + 200 + 1 + 1 + 1) + (1 + 1 + 1 + 1 + \cancel{1+1+1+1}) \times 48$

• The SRAM chunks are hidden from programmer and known as 'Cache'.

30/10

→ Bandwidth of Memory :-

$$\frac{\text{Data}}{\text{sec}} = \frac{\text{Data}}{\text{Instruction}} \times \frac{\text{Instruction}}{\text{Cycle}} \times \frac{\text{Cycle}}{\text{Sec}}$$

$$= \left(\frac{\#\text{Bytes}}{\text{Instruction}} + \frac{\#\text{Bytes of data}}{\text{Instruction}} \right) \times \text{IPC} \times \text{freq}$$

$$= (4B + 4B) \times 1 \times 3 \times 10^9$$

$$\approx 24 \text{ GBps}$$

→ Final CPI depends on no. of 'stalls' required by the program
- Instructions may stall, for reading from or writing to memory, for 200 cycles
- Stalling also happens to fetch instructions not present in cache

• Instructions can be 'hit' or 'miss'

$$\begin{tikzcd} & \text{'hit'} \arrow[dl] & \text{'miss'} \arrow[dr] & \\ \text{present in cache} & & & \text{not present} \\ \text{No penalty} & & & \text{200 cycle penalty} \end{tikzcd}$$

• No. of stalls = No. of misses × penalty

$$= \#\text{Instruction} \times \frac{\#\text{miss}}{\text{Instruction}} \times \text{penalty}$$

$$= \#\text{Inst} \times \frac{\#\text{Memory references}}{\text{Inst}} \times \frac{\#\text{miss}}{\#\text{Mem ref}} \times \text{penalty}$$

$$= \underset{(IC)}{\text{Inst Count}} \times \frac{\#\text{Mem ref}}{\text{Inst}} \times \underbrace{\text{Miss rate} \times \text{penalty}}_{\text{In the hands of memory designer}}$$

→ Miss Rate reduction

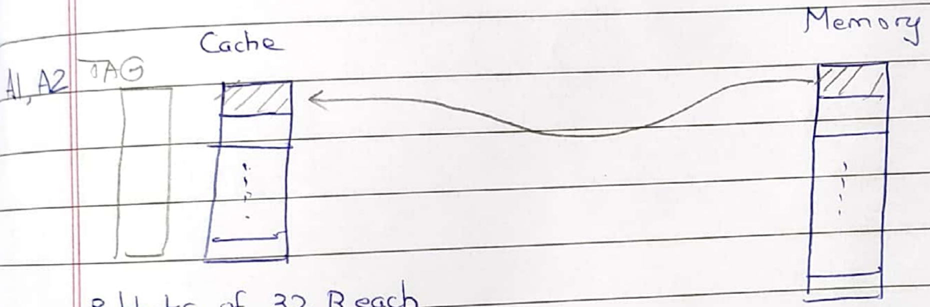
1 Increase no. of blocks in SRAM.

2 Replacing contents of cache :-
Preferentially, do not remove instructions that are consecutively likely to follow current instruction (spatial likeliness) or that are likely to be executed again at some later point in time (temporal likeliness)

classmate
Date
Page

• Increasing the number of blocks in SRAM beyond a certain point will increase access time from 1 cycle to 2 cycles. :(

→ Questions to be answered
1 In which block of cache to place?
2 How to identify where required instruction is placed in cache?
3 Which ~~Whom~~ & cache instruction to replace when overwriting?
4 When to write ~~to~~ ~~cache~~ & data (obtained from processor) from cache to memory?



8 blocks of 32 B each

1024 blocks of 32B each

In memory, every byte can be addressed as

cache

$\underbrace{\boxed{\text{Block No.}}}_{\substack{10 \text{ bits (mem)} \\ 3 \text{ bits (cache)}}}\underbrace{\boxed{\text{Offset}}}_{5 \text{ bits}}$

• We will only be transferring blocks as a whole from memory to cache. After transferring, offset part of each byte remains same. Block no. will be changed accordingly
– We maintain a mapping of which block no. of memory is ~~map~~ transferred to which block no. of cache

– Simplest Mapping = 'Direct Mapping'

(Mem Block No.) % 8 = (Cache Block No.)

eg – Mem Block 45 is transferred to Cache block 5.

P.T.O.

i.e – ~~In~~ ~~Mem bloc~~
(Cache block no.) = Last 3 bits of (Mem block no.)

∴ (Cache address) = Last 8 bits of (Mem address)

– The first seven bits of Mem address (which were lost in transferring to cache) will be stored in an adjacent array to cache. (Tag)
  - TAG is an 8-element array of 7-bit elements each (one element for each block).

T