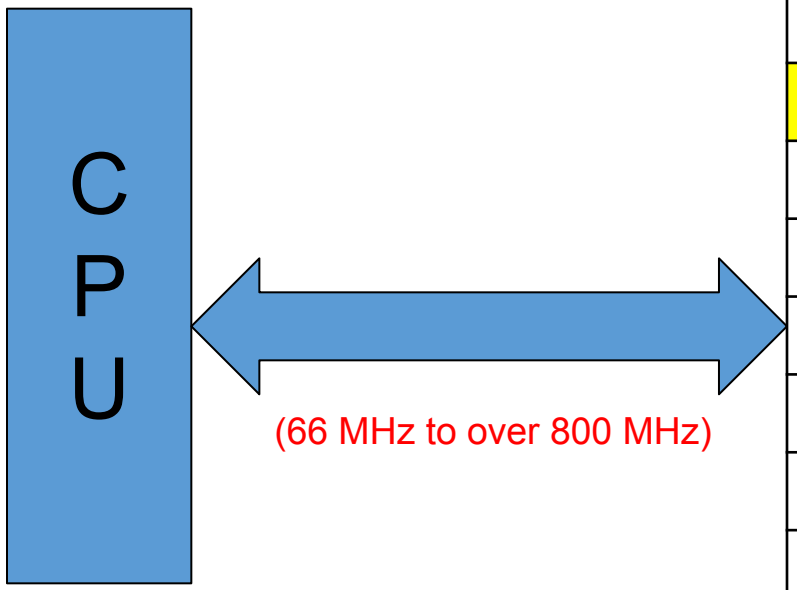
Cache Memory

Simple Model



(clock speed
3.5 **GHz** to 4.0 **GHz)**

	RAM
Address	Contents
125H	Instruction-1
126H	Instruction-2
127H	Instruction-3
128H	Instruction-4
129H	Instruction-5
130H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

User Program

Usually CPU runs Programs sequentially

Data

DRAM speed 1333MHz – 2400MHz

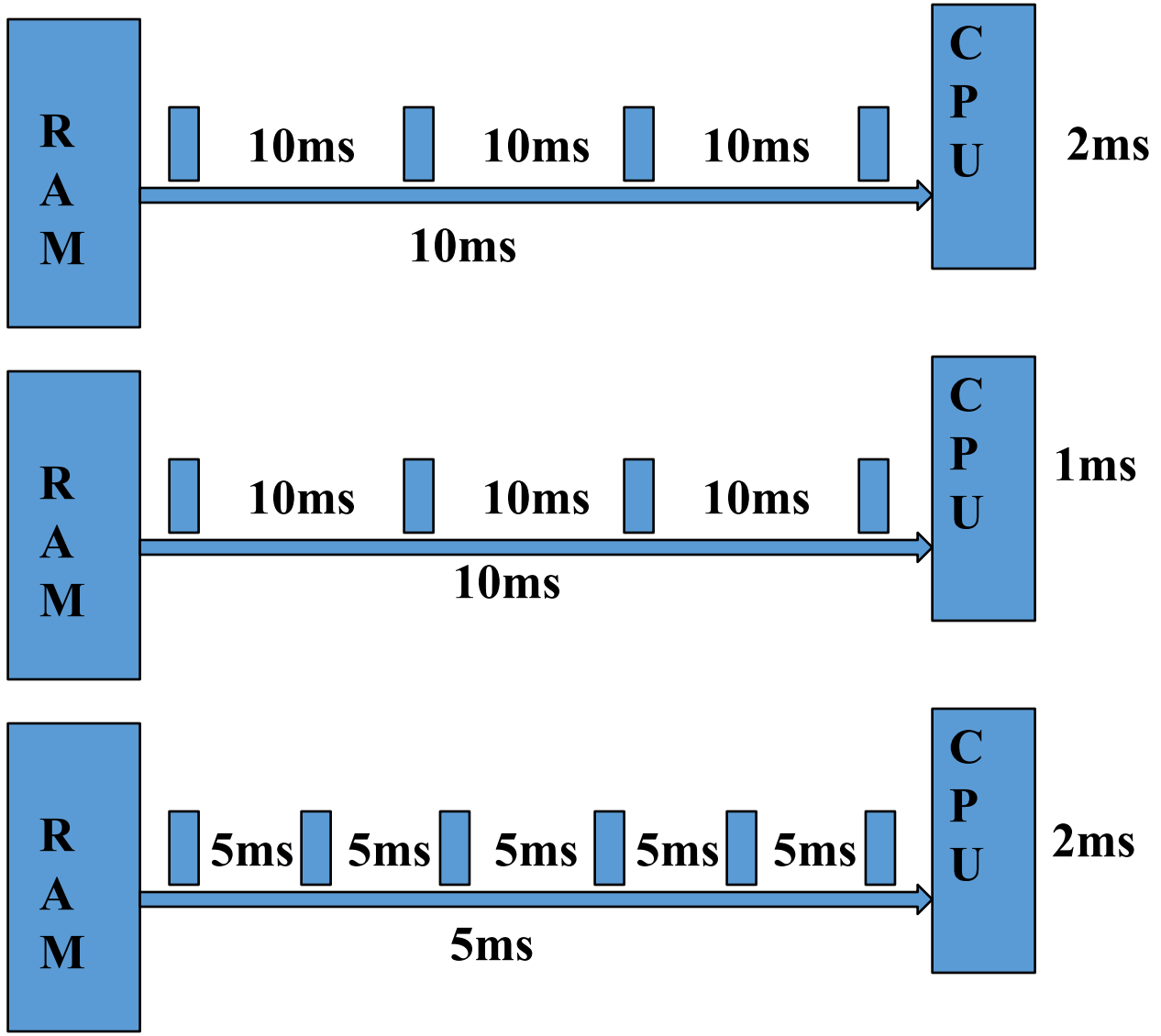
Some models: Processor speed vs Memory and BUS

If RAM & BUS is slower than CPU as it is due to technology and other issues, CPU will remain in wait state most of the time while Instructions are fetched and in case of load/store operations which are very frequent operations.

The raw speed of the microprocessor will not achieve its potential unless it is fed a constant stream of work to do in the form of computer instructions.

To improve Utilization of a processor….

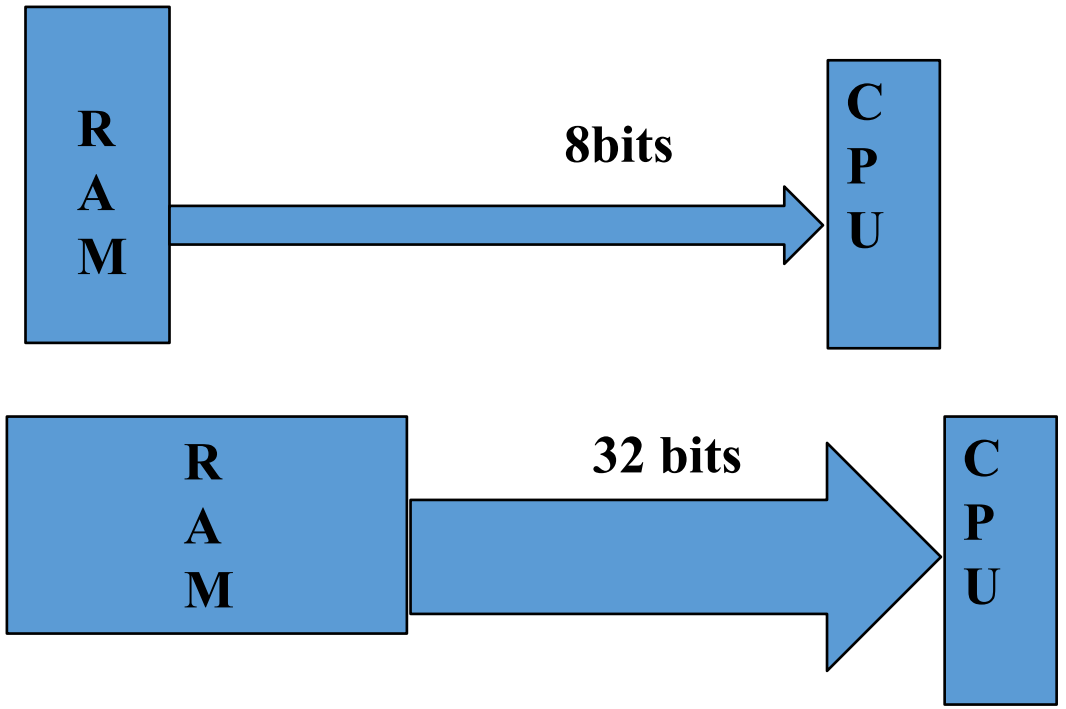
techniques for maintaining a continuous stream of Instructions to the Microprocessor/CPU should be developed and implemented.



Solution: Improved architecture

- Making RAM wider
- Making data bus wider

To allow processor to read more data and instruction in one bus cycle

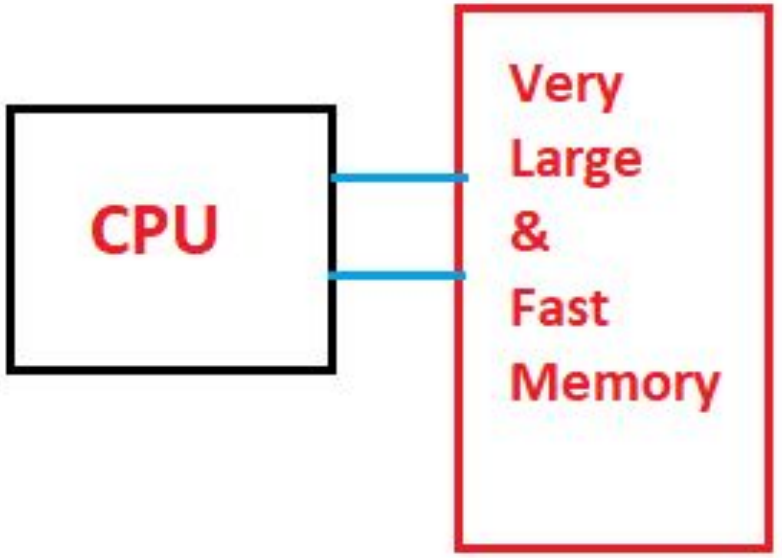


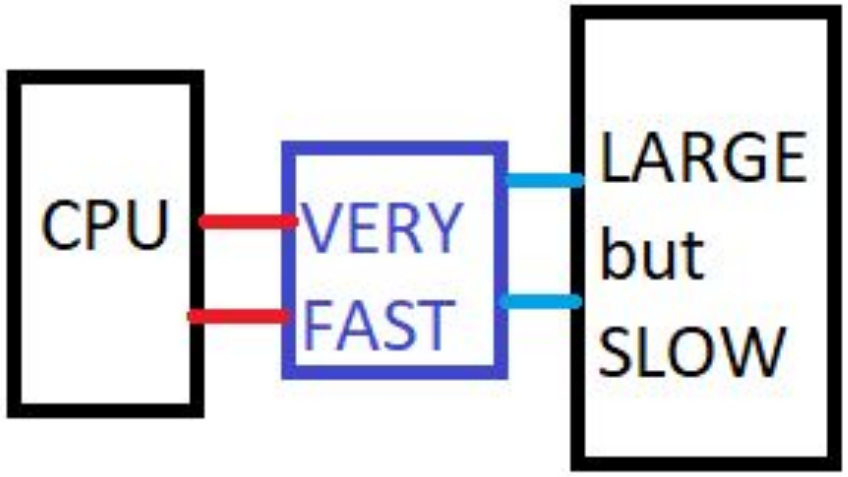
Solution: Memory and BUS speed should be same as Processor speed…consequence…EXPENSIVE!

Programmers/users want memory to be fast and large but CHEAP as well.

Do we need a very fast and large memory?

Since programs access a small proportion of their address space at any time!





Solution: hierarchy of memories to optimize cost and speed.

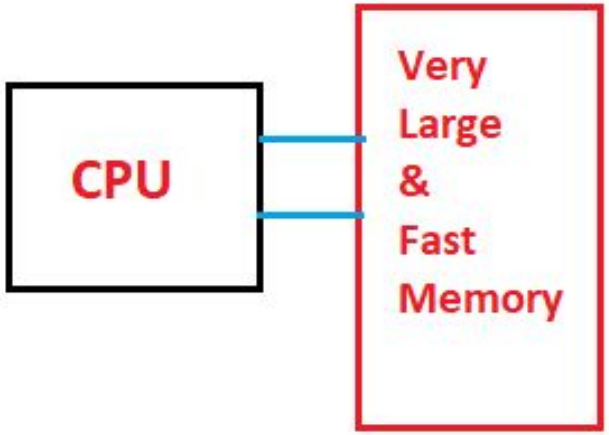
Memory Hierarchy

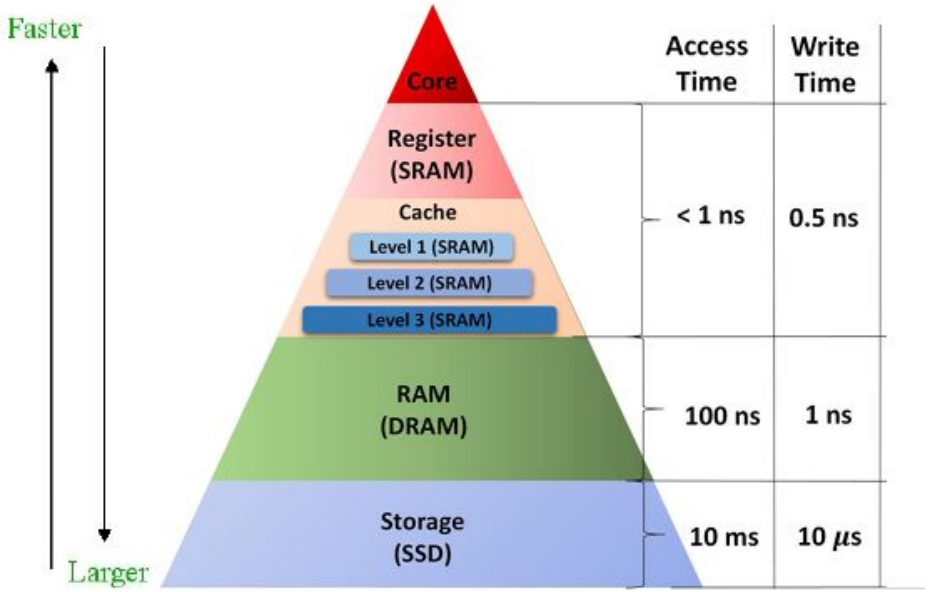
- Use combination of memory kinds
  - Smaller amounts of expensive but fast memory closer to the processor
  - Larger amounts of cheaper but slower memory farther from the processor
- Idea is not new:

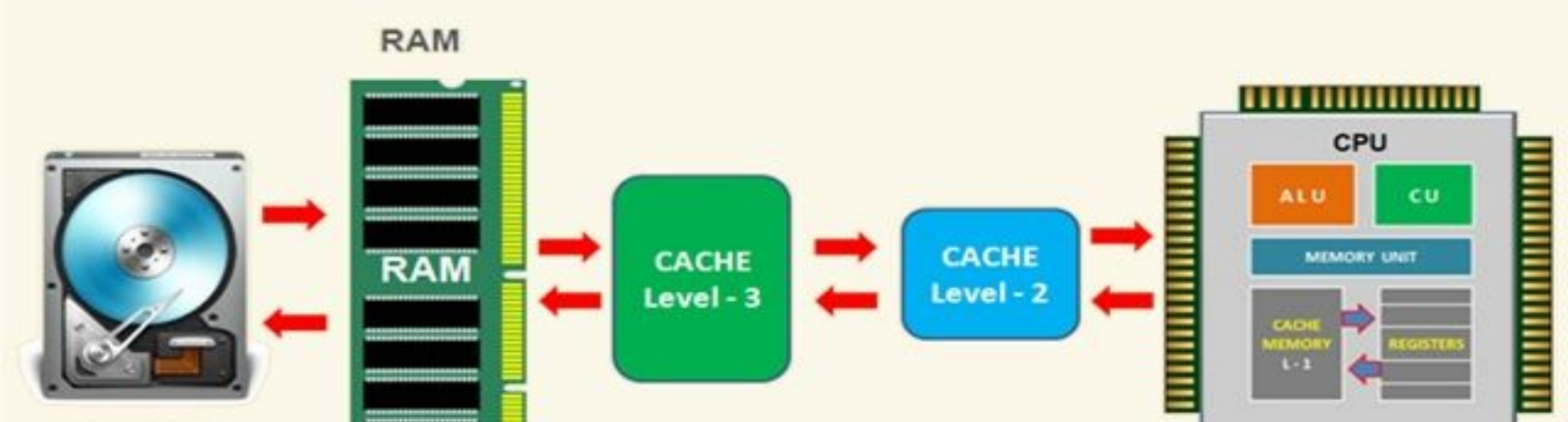
"Ideally one would desire an indefinitely large memory capacity such that any particular … word would be immediately available… we are … forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."

A. W. Burks, H. H. Goldstine, and J. von Neumann - 1946

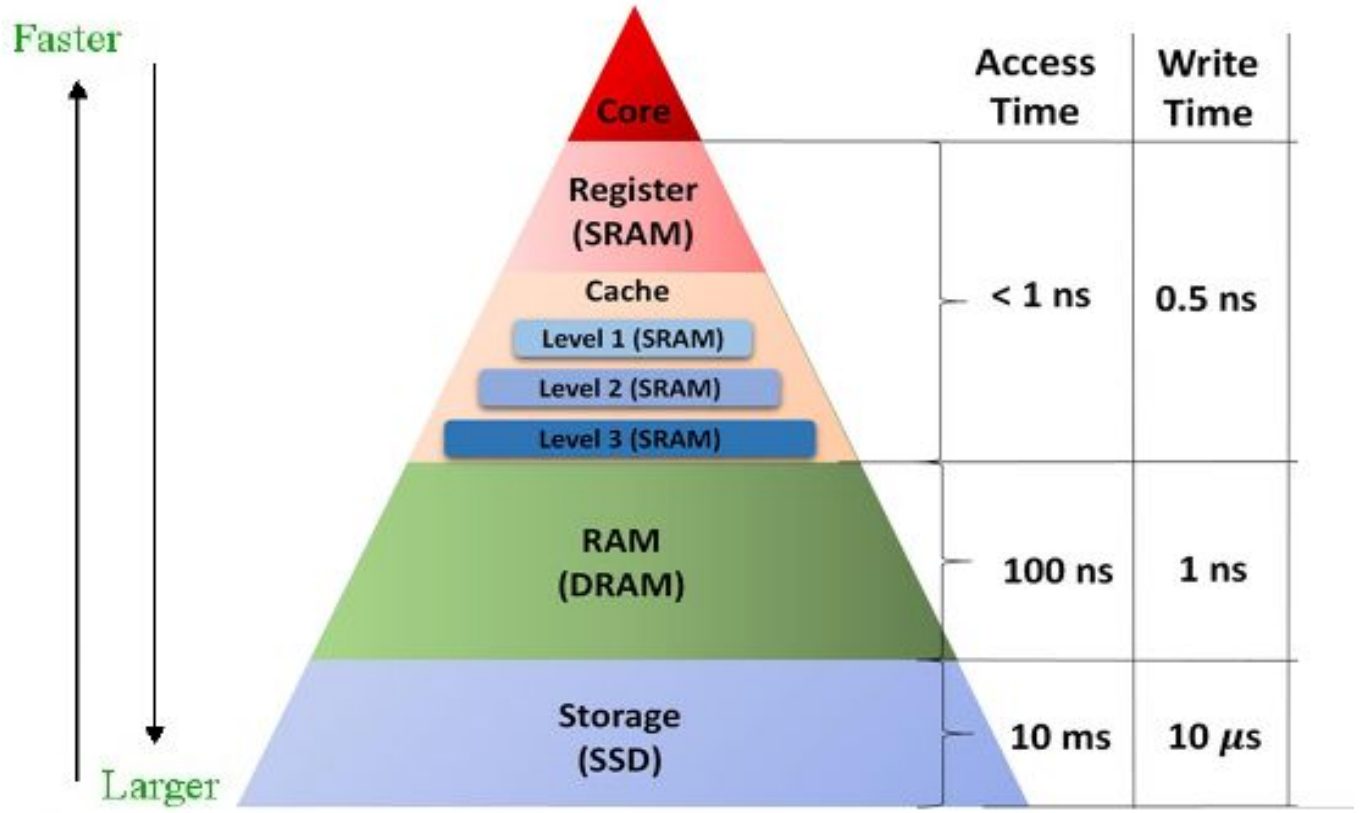
Why Hierarchy of Memories? To optimize cost and speed.
look ≈as fast as most expensive memory, ≈ as big as cheapest





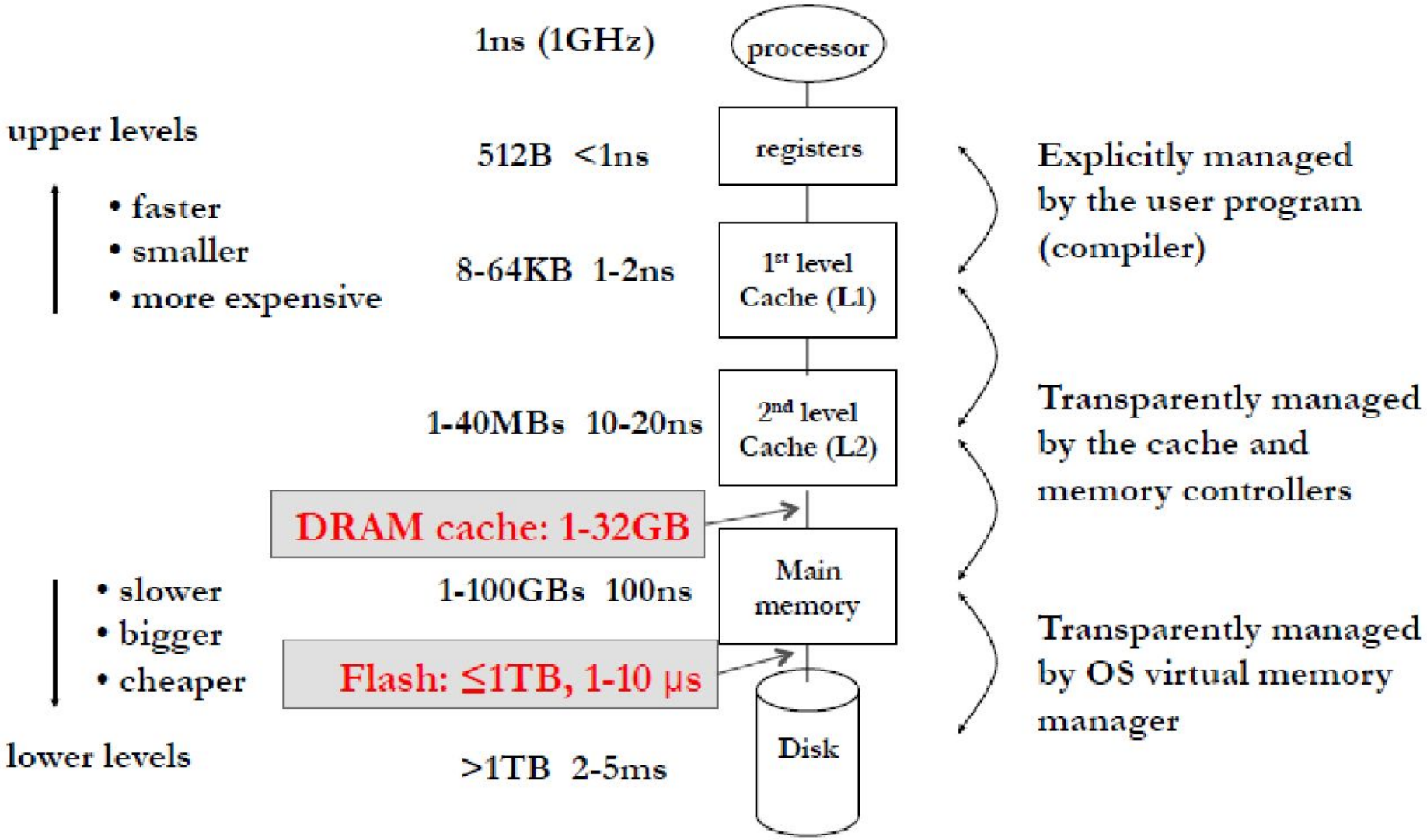


Memory Hierarchy: Optimize Cost and Access Time



The fastest, smallest, and most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom.

Memory Hierarchy



Program and programming

- Instruction Reuse: Loops and Functions
- Data working set: Arrays

- Programs access a small proportion of their address space at any time

(locality of reference)

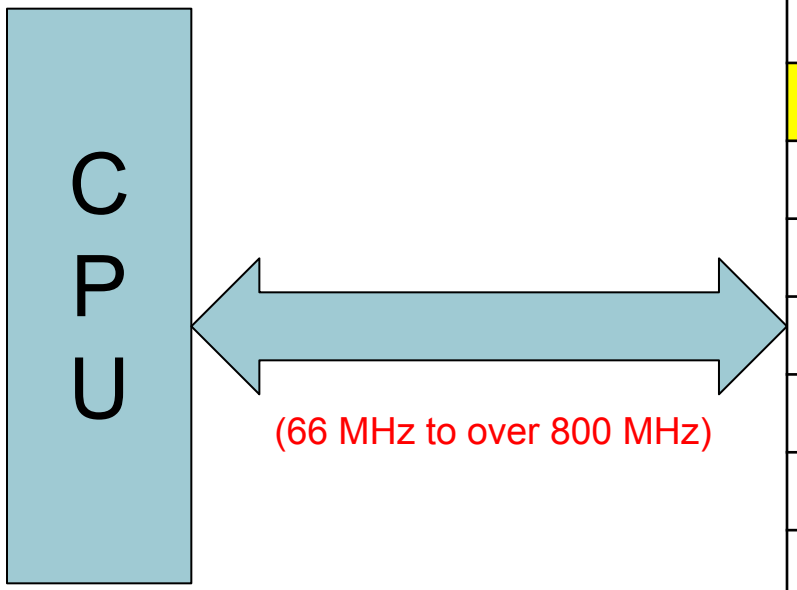
Principle of Locality

- Programs access a small proportion of their address space at any time

- Temporal locality
  - Items accessed recently are likely to be accessed again soon
  - instructions in a loop, induction variables

- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - sequential instruction access, array data

Simple Model



(clock speed
3.5 **GHz** to 4.0 **GHz)**

	RAM
Address	Contents
125H	Instruction-1
126H	Instruction-2
127H	Instruction-3
128H	Instruction-4
129H	Instruction-5
130H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

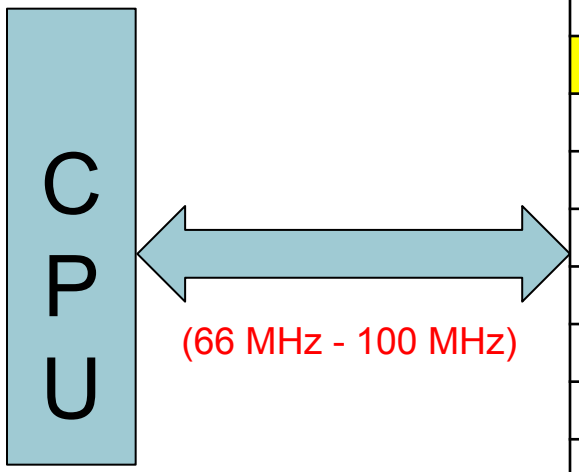
User Program

Usually CPU runs Programs sequentially

Data

DRAM speed 1333MHz – 2400MHz

Simple Model



(66 MHz - 100 MHz)

(clock speed
3.5 **GHz** to 4.0 **GHz)**

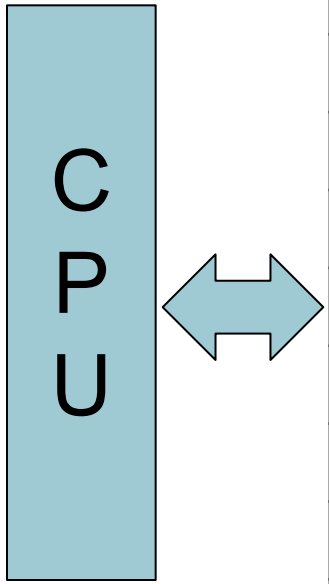
	RAM
Address	Contents
125H	Instruction-1
126H	Instruction-2
127H	Instruction-3
128H	Instruction-4
129H	Instruction-5
130H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

1333MHz – 2400MHz

- RAM contains program and data
- A single very large and fast memory could have been better
- But CPU can process single or a few instructions at a time
- CPU accesses a small proportion of RAM at any time
- Do we need- a single very large and fast memory?

Next Instruction??



	RAM
Address	Contents
125H	Instruction-1
126H	Instruction-2
127H	Instruction-3
128H	Instruction-4
129H	Instruction-5
130H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

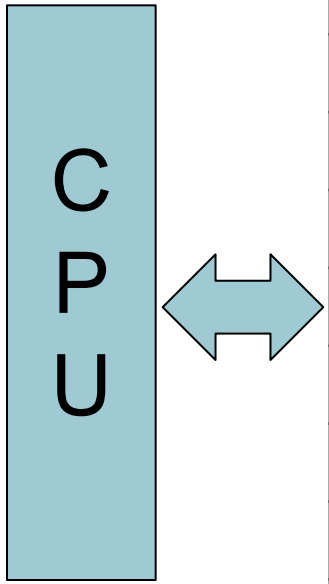
Currently CPU executes Instruction-1

Next Instruction??

Instruction-2, 3 (likely) Stored at Memory locations (126H, 127…) are likely to be accessed

Instruction-1 could be repeated in case of loop

Locality of Reference: Next Instruction??



	RAM
Address	Contents
125H	Instruction-1
126H	Instruction-2
127H	Instruction-3
128H	Instruction-4
129H	Instruction-5
130H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

Currently
CPU executes
Instruction-1

Next Instruction??

Temporal locality

Recently accessed memory location 125H is likely to be accessed again
Instruction-1 repeat

Spatial locality

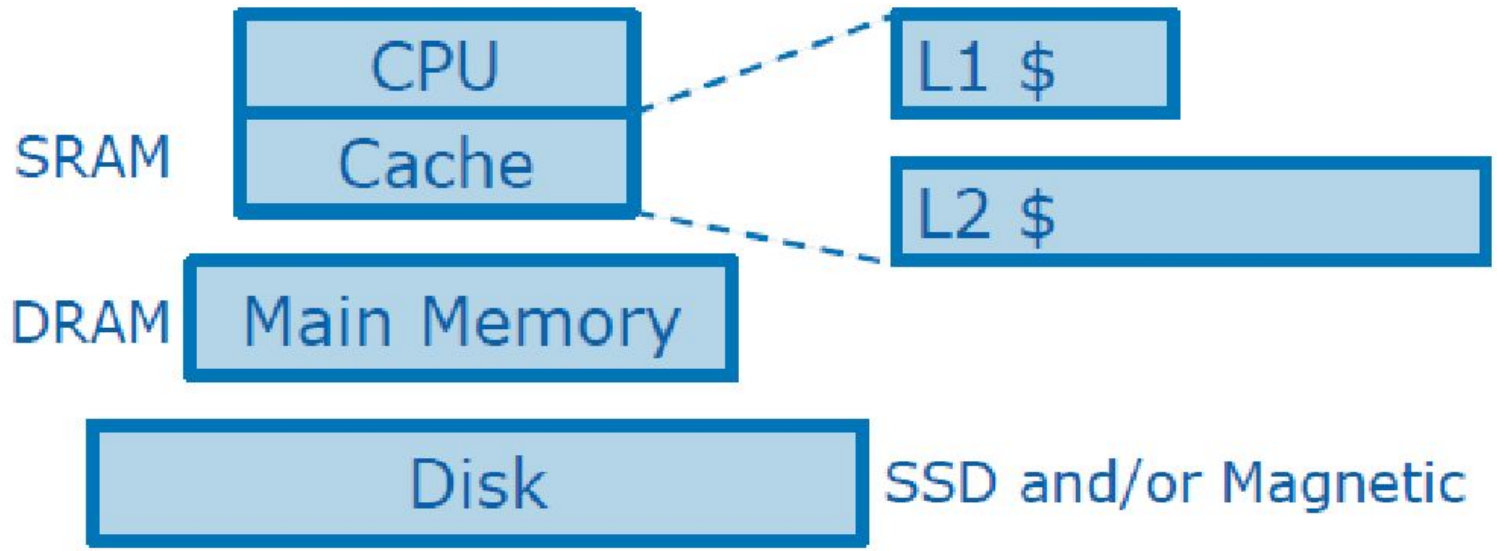
Memory locations close to recently accessed are likely to be accessed
Instruction-2, 3 (likely)
Stored at 126H, 127H

Taking Advantage of Locality

- Memory hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU

Taking Advantage of Locality

- **Memory hierarchy**



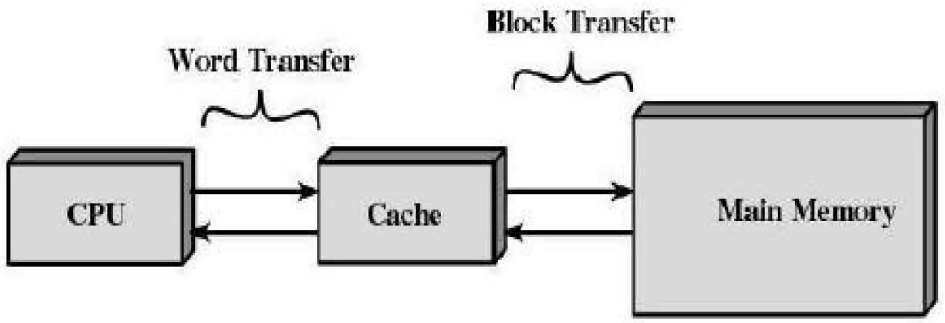
- **Store everything on disk**
- **Copy recently accessed (and nearby) items from disk to smaller DRAM memory**
- **Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory**

Hierarchy of memories and Cache Memory

- Programmers want memory to be fast, large, and cheap, as memory speed often shapes performance, capacity limits the size of problems that can be solved, and the cost of memory today is often the majority of computer cost. Architects have found that they can address these conflicting demands with a hierarchy of memories, with the fastest, smallest, and most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom. Caches give the programmer the illusion that main memory is nearly as fast as the top of the hierarchy and nearly as big and cheap as the bottom of the hierarchy.

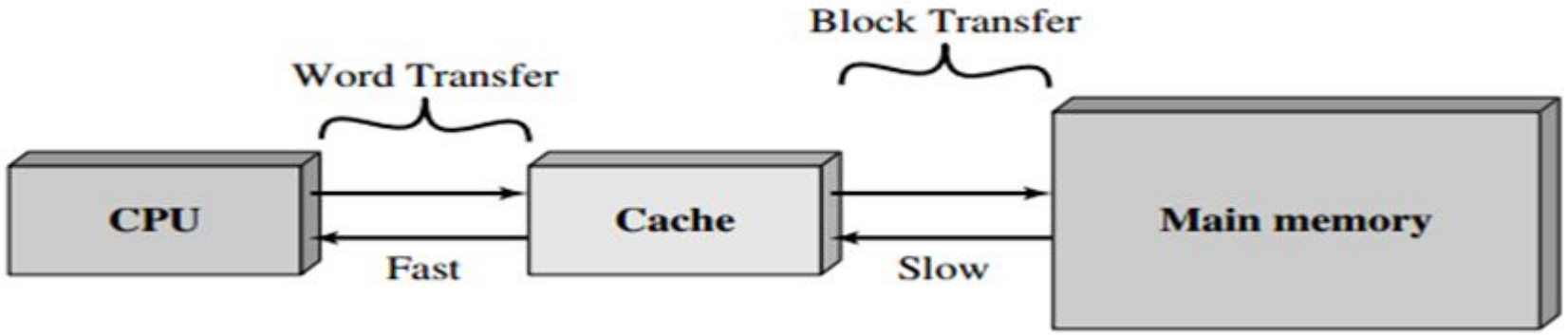
Example

- A program contains 1000 Instructions (machine) each of 1 byte length. Processor-A does not contain any cache memory and it can read 1 byte instruction in a single read operation. (All instructions are register based, example: Add R4, R2, R3)
- How many memory access would require to run the program?
  Answer……?
- A program contains 1000 Instructions (machine) each of 1 byte length. Level-1 cache is used as shown and CPU can find/read 900 instructions from that. (Hit ratio = (900/1000)x 100 = 90%).
- How many memory access would require to run the program?
  Answer……?

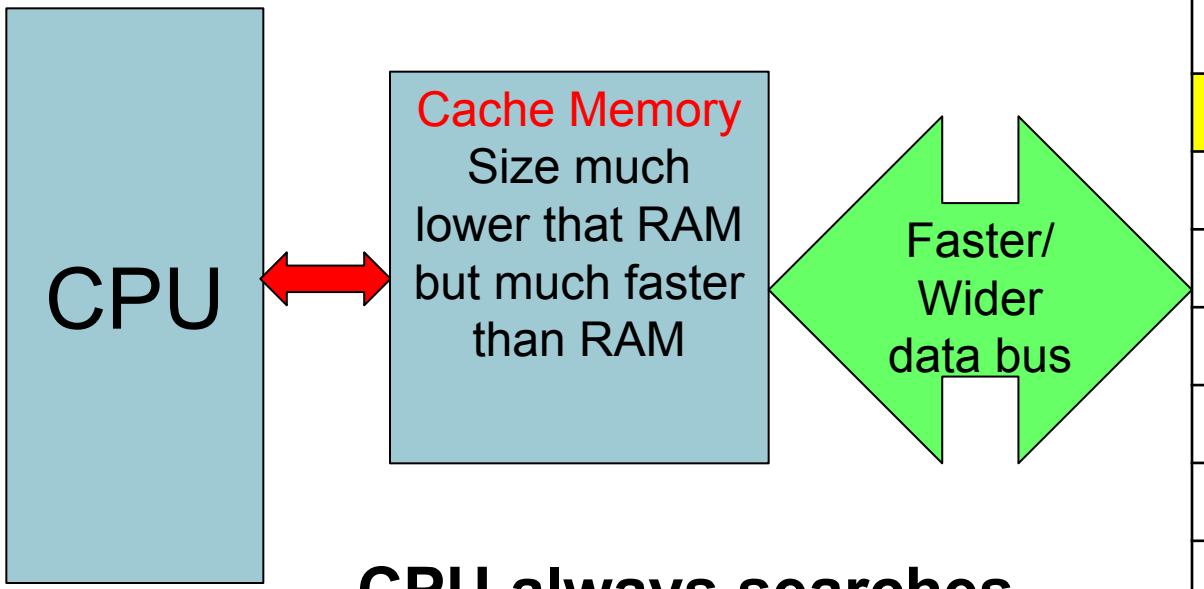


Cache Memory:
Reduces the frequency of memory access

- Cache Memory: Reduce the frequency of memory access
- Reduces average memory access time of a program
- Reduces program execution time
- Caches give the programmer the illusion that main memory is nearly as fast as the top of the hierarchy and nearly as big and cheap as the bottom of the hierarchy.



Cache Memory reduces frequency of access to RAM while CPU is running a program



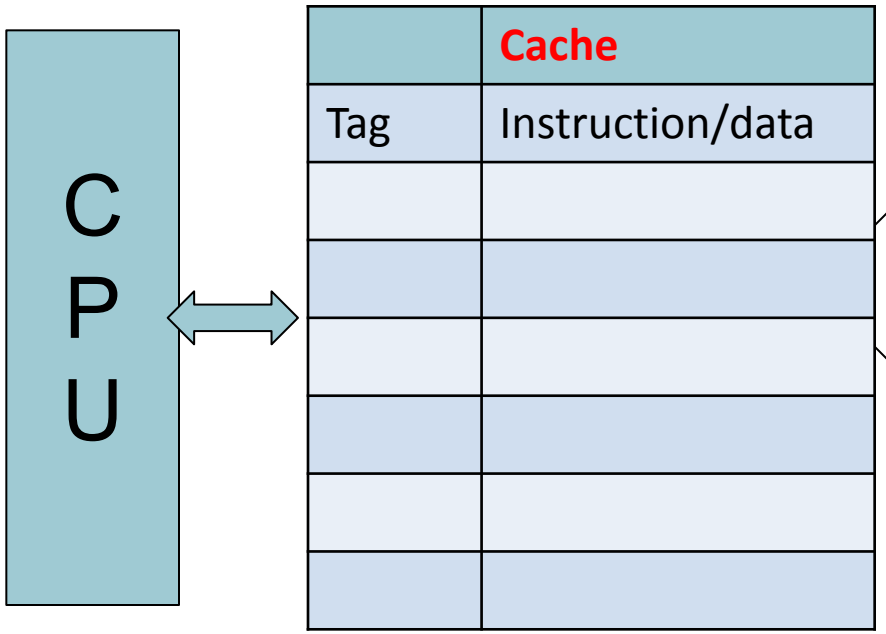
CPU always searches next instruction/data in cache first by its memory(RAM) address

	RAM
Address	Contents
120H	Instruction-1
121H	Instruction-2
122H	Instruction-3
123H	Instruction-4
124H	Instruction-5
135H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

Cache Memory reduces frequency of access to RAM while CPU is running a program

Initially Cache Empty



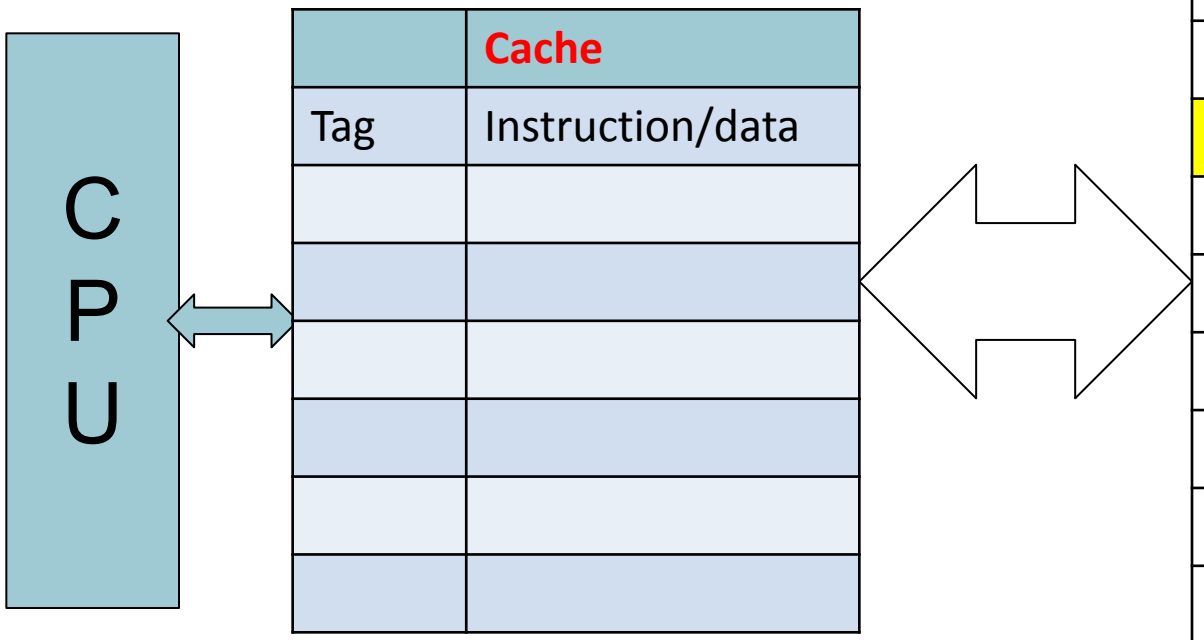
	RAM
Address	Contents
120H	Instruction-1
121H	Instruction-2
122H	Instruction-3
123H	Instruction-4
124H	Instruction-5
135H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

CPU will search instruction-1 in cache first by its memory address 120H, it means, whether contents of 120H are available in cache

Cache Memory reduces frequency of access to RAM while CPU is running a program

Initially Cache Empty

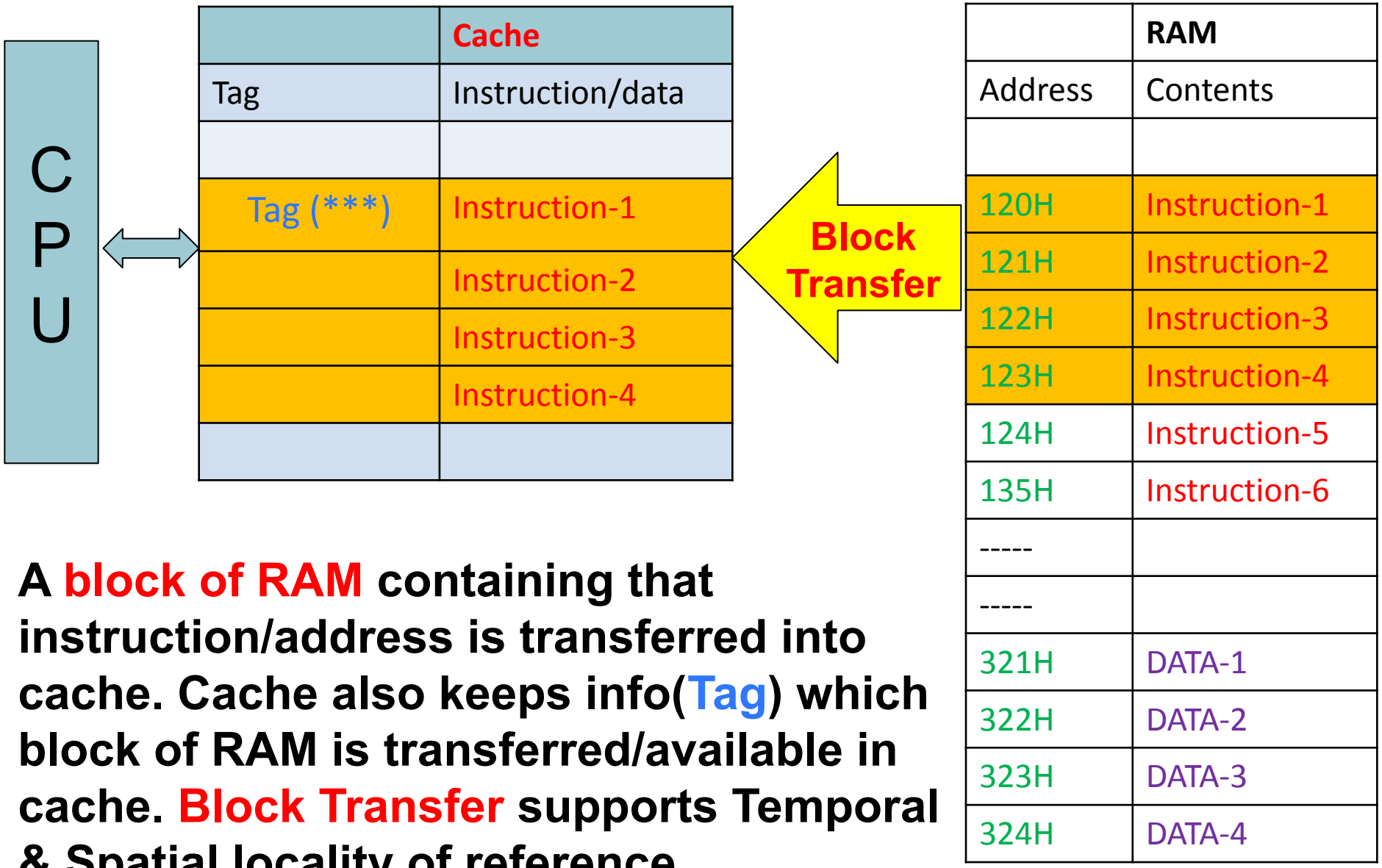


	RAM
Address	Contents
120H	Instruction-1
121H	Instruction-2
122H	Instruction-3
123H	Instruction-4
124H	Instruction-5
135H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

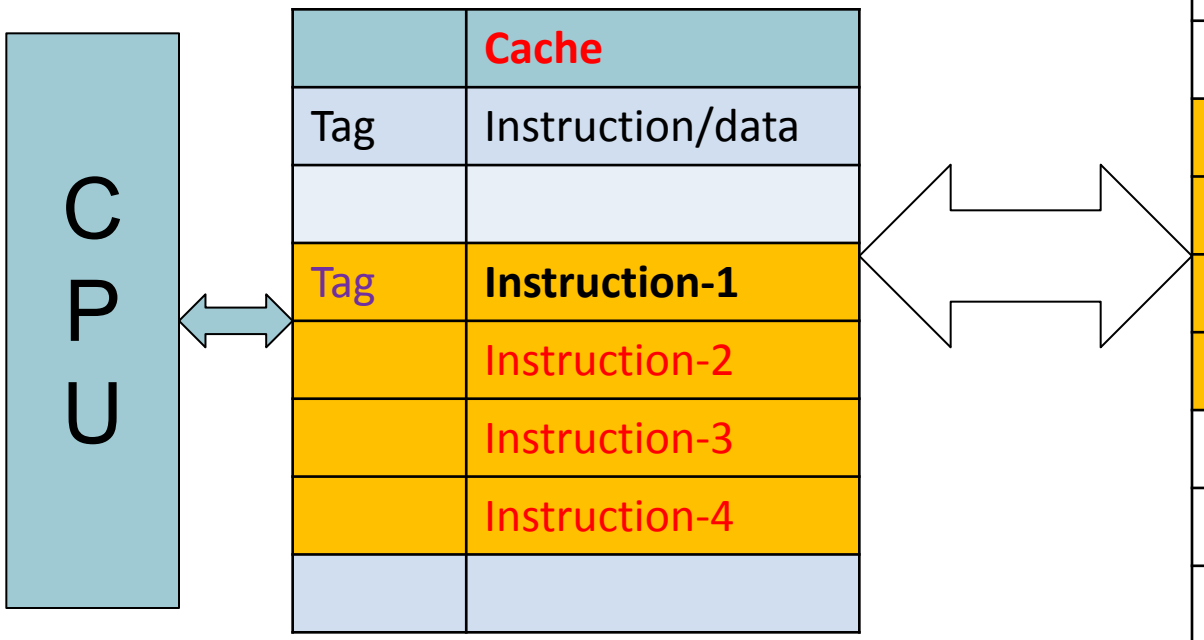
Since Cache is Empty CPU doesn't find instruction-1 in cache, i.e., address 120H is not found in cache, CACHE MISS

Cache Memory reduces frequency of access to RAM by using the concept of Locality of Reference



A block of RAM containing that instruction/address is transferred into cache. Cache also keeps info(Tag) which block of RAM is transferred/available in cache. Block Transfer supports Temporal & Spatial locality of reference

Cache Memory reduces frequency of access to RAM by using concept of Locality of Reference

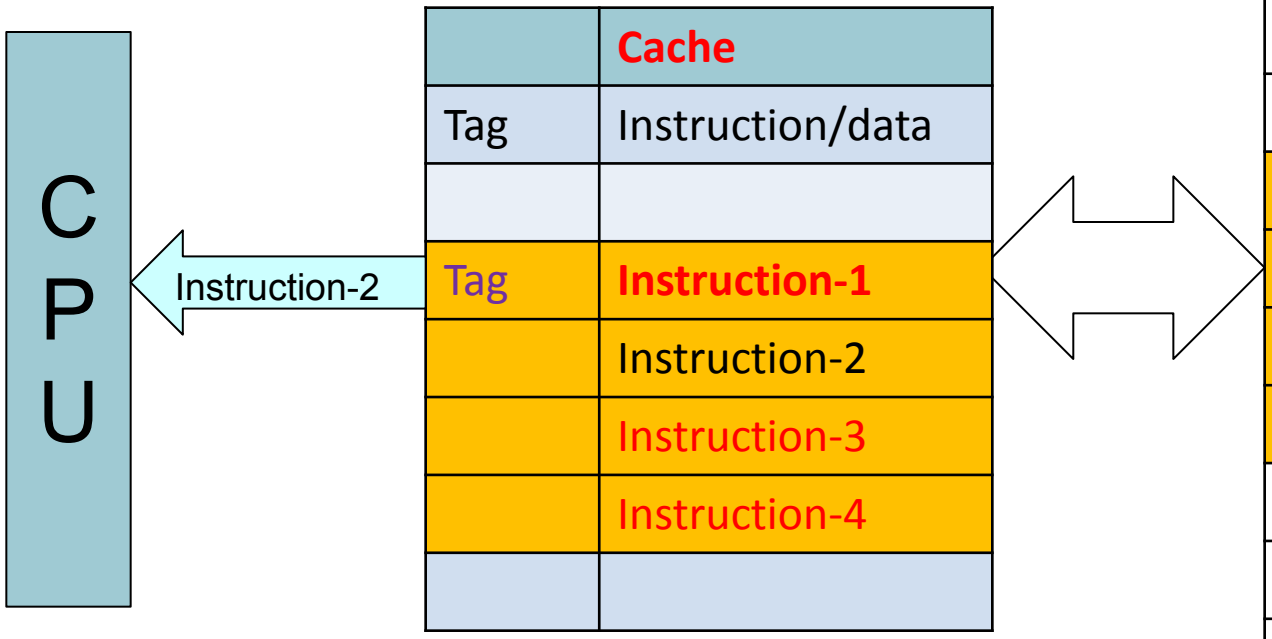


	RAM
Address	Contents
120H	Instruction-1
121H	Instruction-2
122H	Instruction-3
123H	Instruction-4
124H	Instruction-5
135H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

CPU reads first instruction (Instruction-1) from cache and executes it.

Cache Memory reduces frequency of access to RAM by using concept of Locality of Reference

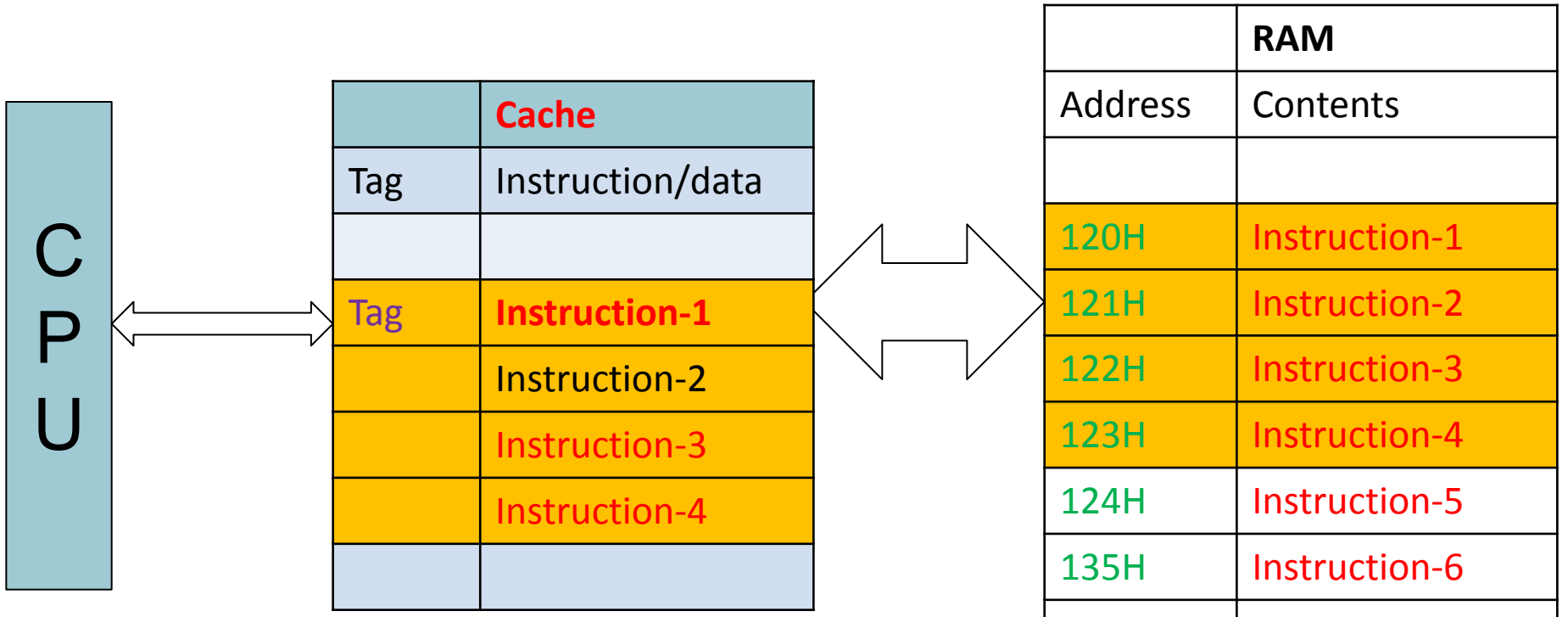


	RAM
Address	Contents
120H	Instruction-1
121H	Instruction-2
122H	Instruction-3
123H	Instruction-4
124H	Instruction-5
135H	Instruction-6

321H	DATA-1
322H	DATA-2
323H	DATA-3
324H	DATA-4

Likewise CPU will search next instruction, instruction-2 in cache first by its memory address 121H, and finds it, called CACHE HIT.
CPU simply reads Instruction-2 from cache, no need to access RAM!

Cache Memory reduces frequency of access to RAM by using concept of Locality of Reference

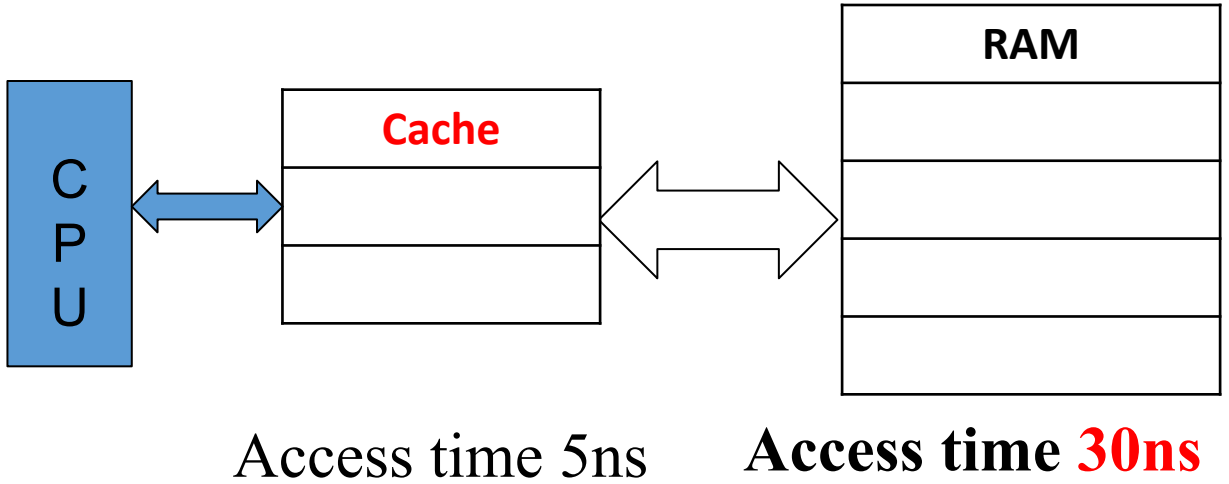


Likewise CPU will continue to read following instructions from cache, unless there is a Cache Miss. In that case, a block of RAM containing that address is transferred into cache.

Hit Ratio

- Suppose a program, initially loaded into RAM, contains 100 instructions (all instructions are register based). When it is run, CPU reads/finds 80 instructions from cache memory. For the remaining instructions, CPU has to access RAM.
- In such case, the Hit ratio of Cache for the program would be 80%, means that 80 out of 100 instructions are found in cache.
- Suppose a program, initially loaded into RAM, contains 300 instructions (all instructions are register based). When it is run, CPU reads/finds 250 instruction from cache memory. For the remaining instructions, CPU has to access RAM. Calculate Hit Ratio: (250/300)x 100 = 83.33%
- What would be the Miss Rate?

- Cache Memory Reduce the frequency of memory access
- Cache memory reduces average memory access time of a program.



Suppose a program, initially loaded into RAM, contains 100 instructions (all instructions are register based). When it is run, CPU reads/finds 80 instruction from cache memory. For the remaining instructions, CPU has to access RAM. Calculate average access time.

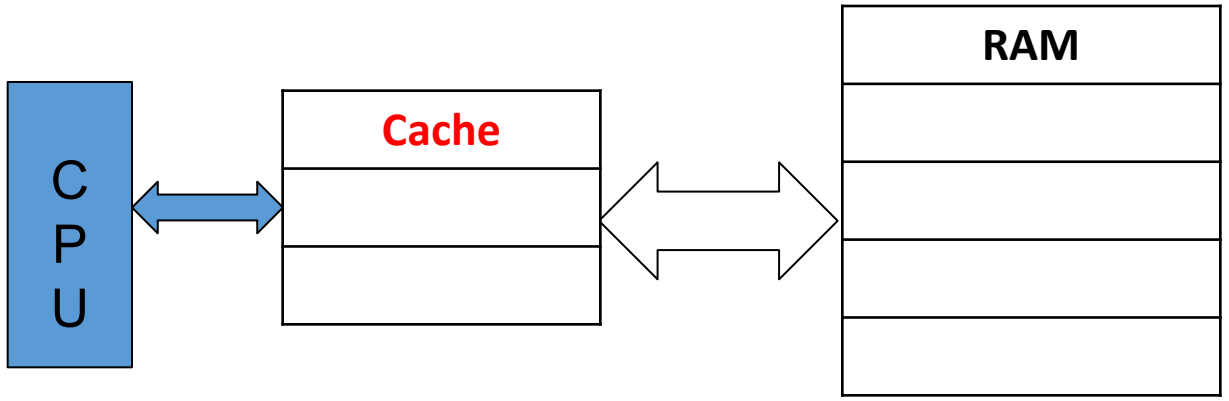
Solution:

Total time to read 80 instructions from cache: 80×5ns = 400ns

Total time to read 20 instructions from RAM: 20×(30ns+5ns) = 700ns

Total access time for the program: 400ns + 700ns = 1100ns

Average access time: 1100ns/100 = 11ns



Suppose a program, initially loaded into RAM, contains 100 instructions (all instructions are register based). The Hit ratio of Cache is 80%. Calculate average access time.

Solution:

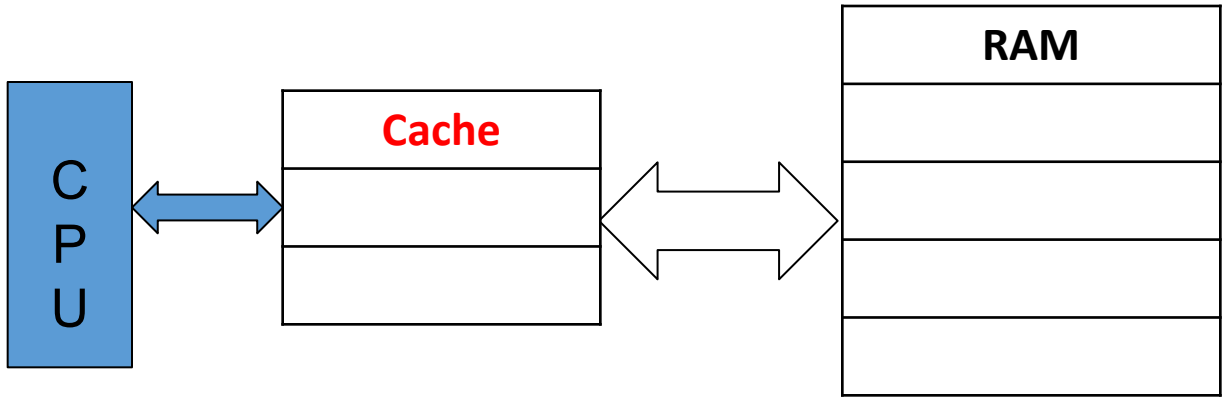
Total Instructions = 100 and 80 instructions are read from cache.

Total time to read 80 instructions from cache: 80×5ns = 400ns

Total time to read 20 instructions from RAM: 20×(30ns+5ns) = 700ns

Total access time for the program: 400ns + 700ns = 1100ns

Average access time: 1100ns/100 = 11ns



Access time 5ns

Access time 30ns

Suppose a program(all instructions are register based), initially loaded into RAM. The Hit ratio of Cache is 80%. Calculate average access time.

Solution:

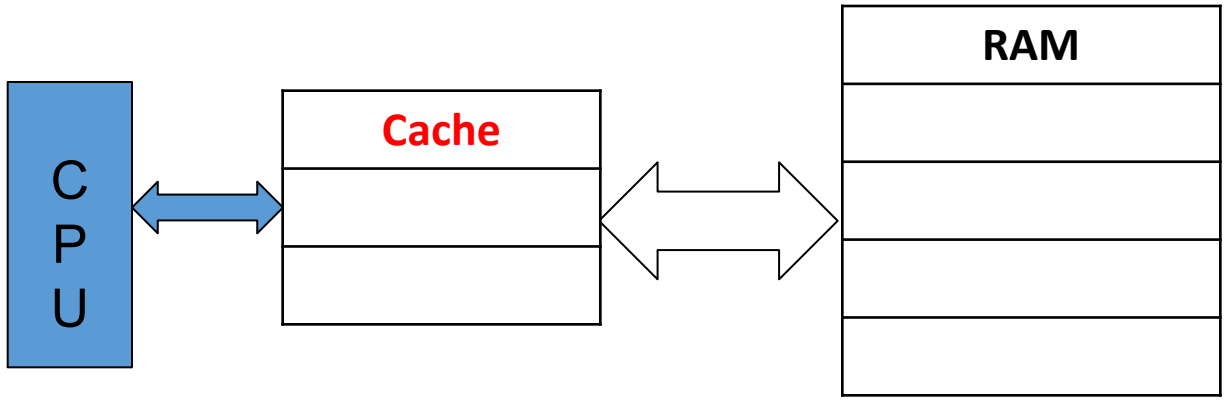
80 out of 100 instructions are read from cache.

Total time to read 80 instructions from cache: 80×5ns = 400ns

Total time to read 20 instructions from RAM: 20×(30ns+5ns) = 700ns

Total access time for the program: 400ns + 700ns = 1100ns

Average access time: 1100ns/100 = 11ns

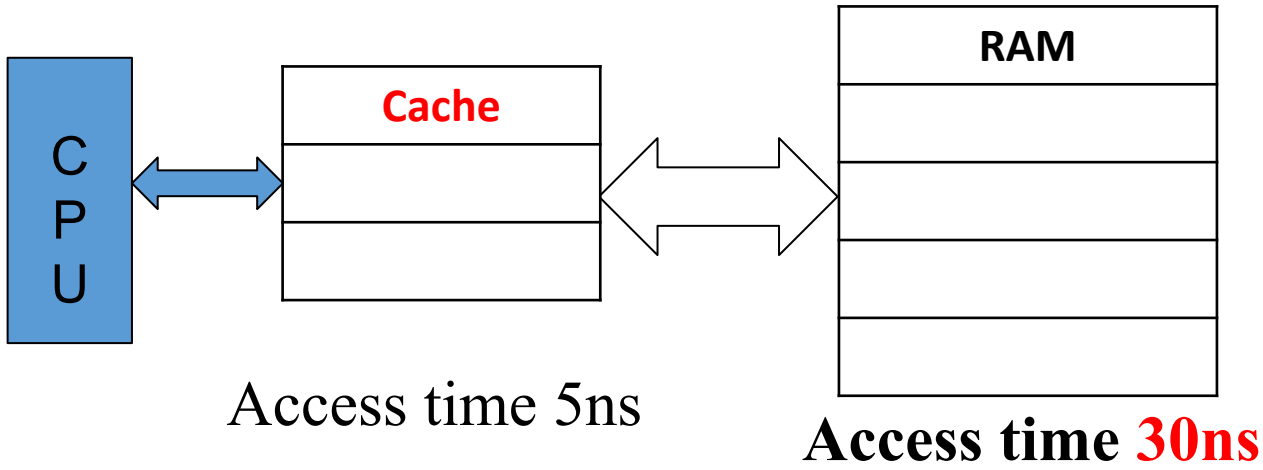


Suppose a program(all instructions are register based), initially loaded into RAM. The Hit ratio of Cache is 80%. Calculate average access time.

Solution:

Average access time: $0.8 \times 5ns + (1 - 0.8) \times 35ns = 11ns$

If Hit ratio of Cache is 80%
Miss Rate = 1- 0.8 = 0.2
Miss Penalty (MP) = 30ns
Hit Time (HT) = 5ns
AMAT = Hit Time + Miss Rate x Miss Penalty
= 5ns +0.2×30ns = 11ns



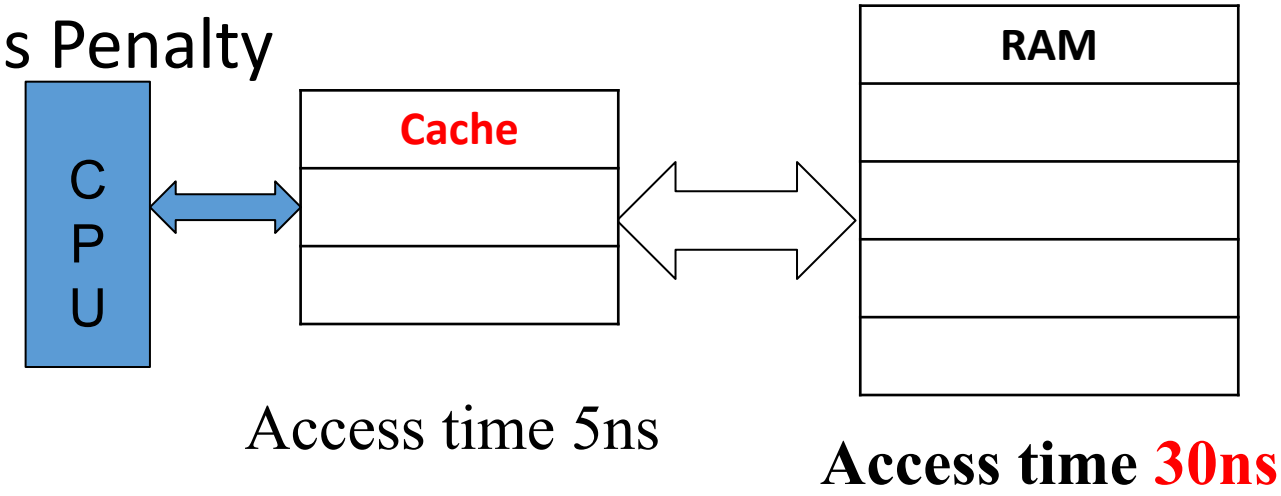
- Hit Time (HT): The hit time is how long it takes data to be sent from the cache to the processor. This is usually fast, on the order of 1-3 clock cycles.
- Miss Penalty (MP): The miss penalty is the time to copy data from main memory to the cache. This often requires dozens of clock cycles (at least). The miss rate is the percentage of misses.
- Miss Rate (MR): 1 – Hit Ratio
- The average memory access time, or AMAT, can then be computed.
  AMAT = Hit time + (Miss rate × Miss penalty)
  This is just averaging the amount of time for cache hits and the amount of time for cache misses

Cache performance measured using AMAT

Parameters that matter:

- Hit Time (HT)
- Miss Rate (MR)
- Miss Penalty (MP)

AMAT = Hit Time + Miss Rate x Miss Penalty



Suppose a program(all instructions are register based), initially loaded into RAM. The Hit ratio of Cache is 80%. Calculate average access time.

Solution: AMAT = Hit Time + Miss Rate x Miss Penalty

Average Memory Access Time: 5ns +0.2×30ns = 11ns

Cache Performance

- Two things hurt the performance of a cache:
  - Miss rate and miss penalty
- *Average Memory Access Time* (AMAT): average time to access memory considering both hits and misses

AMAT = Hit time + Miss rate × Miss penalty

(abbreviated AMAT = HT + MR × MP)

Assume a memory access to main memory on a cache "miss" takes 30 ns and a memory access to the cache on a cache "hit" takes 3 ns. If 80% of the processor's memory requests result in a cache "hit", what is the average memory access time?

AMAT = HT + MRxMP

3ns + 0.2x 30

3ns + 6 = 9

AMAT = 0.8x3 + 0.2 x 33

2.4 + 6.6

9

How can we improve the average memory access time of a system?

- Obviously, a lower AMAT is better. Miss penalties are usually much greater than hit times, so the best way to lower AMAT is to reduce the miss penalty or the miss rate. However, AMAT should only be used as a general guideline. Remember that execution time is still the best performance metric.

Another way of measuring cache performance is by computing the number of cycles the processor must stall for all the memory accesses in a given program.

We can compute this with the following formula:

Memory Stall Cycles = Memory Accesses × Miss rate × Miss penalty

Problem

If a memory system consists of a single external cache with an access time of 20 ns and a hit rate of 0.92, and a main memory with an access time of 60 ns, what is the average memory access time (AMAT) of this system?

AMAT = Hit time + Miss rate × Miss penalty

AMAT = 20 + {0.08x60} = 24.8 ns

Example

- A machine has a base CPI (hit time) of 2 clock cycles. Measurements obtained show that the instruction miss rate is 12% and the data miss rate is 6%, and that on average, 30% of all instructions contain one data reference. The miss penalty for the cache is 10 cycles. What is the Average CPI?

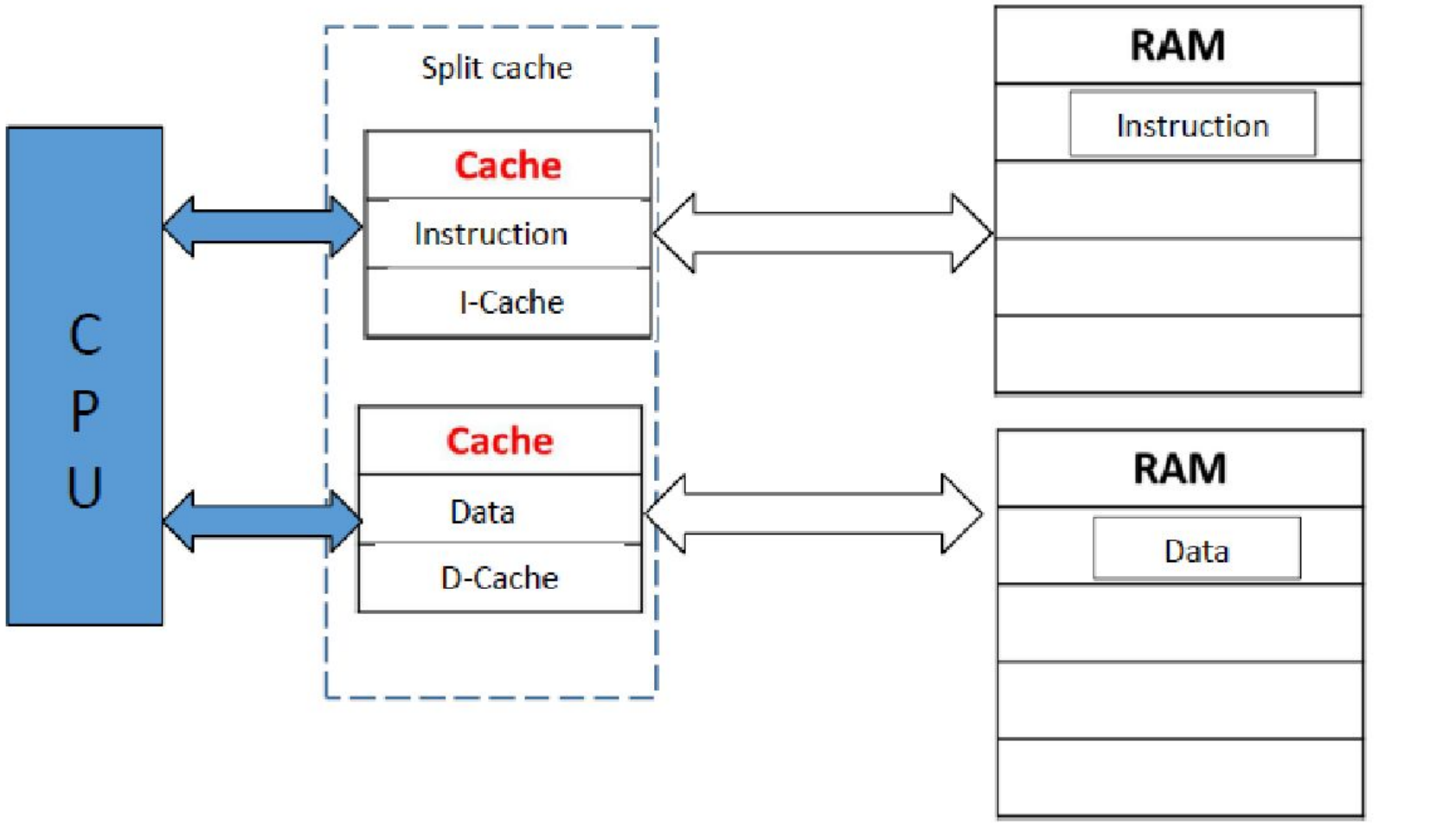
Average CPI =

base CPI (hit time) + instruction miss cycles + data miss cycles

= 2.0 + 0.12x10 + 0.30x0.06x10 = 2.0 + 1.2 + 0.18

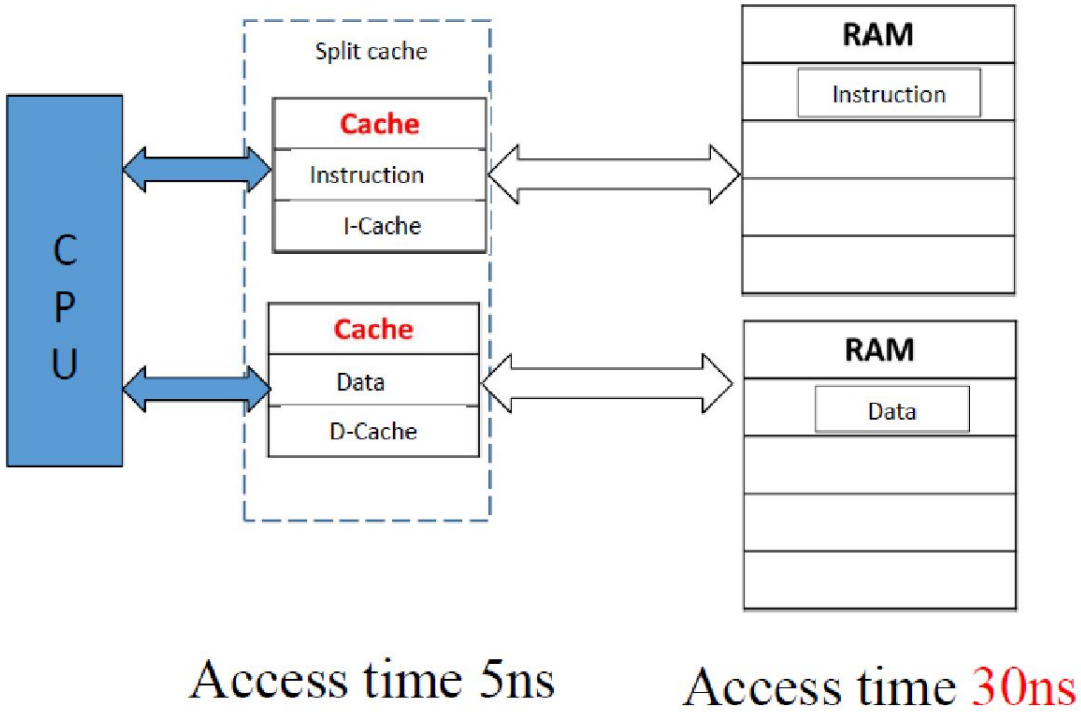
= 3.38

Split cache (Harvard cache)



What advantage does a Harvard cache have over a unified cache?

A Harvard (split) cache permits the processor to access both an instruction word and a data word in a single cache memory cycle. This can significantly increase the performance of the processor. The only real drawback of a Harvard cache is that the processor needs two memory busses (two complete sets of memory address and data lines), one to each of the caches. However, when the Harvard cache is implemented within the processor chip, the separate memory busses never have to escape the chip, being implemented completely in the metal interconnections on the chip.



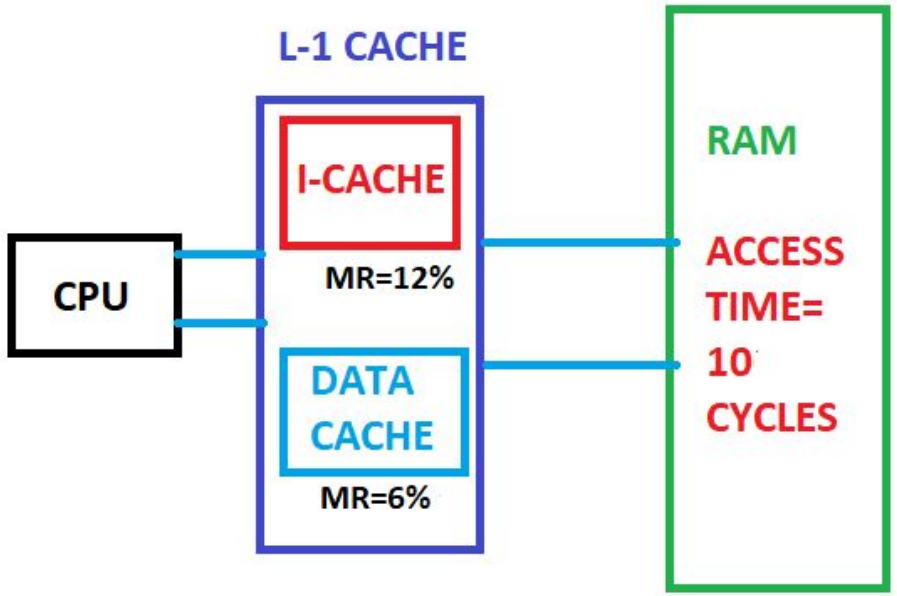
A machine has a base CPI of 2 clock cycles. Measurements obtained show that the instruction miss rate is 12% and the data miss rate is 6%, and that on average, 30% of all instructions contain one data reference. The miss penalty for the cache is 10 cycles. What is the Average CPI?

Average CPI =

2.0 + instruction miss cycles + data miss cycles

= 2.0 + 0.12×10 + 0.30×0.06×10

= 2.0 + 1.2 + 0.18 = 3.38



Machine has a base CPI of 1 clock cycles. Measurements obtained show that the instruction miss rate is 15% and the data miss rate is 6%, and that on average, 40% of all instructions contain one data reference. The miss penalty for the cache is 20 cycles. What is the Average CPI?

Solution:

Average CPI = Base CPI + instruction miss cycles + data miss cycles

= 1 + 0.15 x 20 + 0.40 x 0.06 x 20 = 4.48 clock cycles

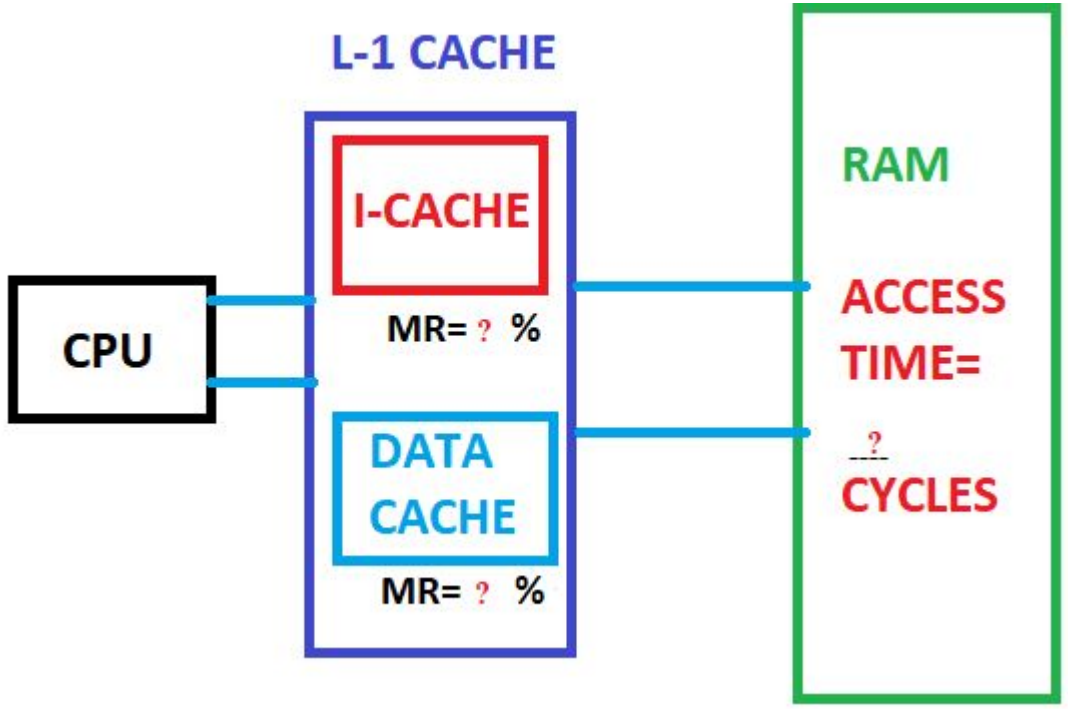
Machine has a base CPI of 1 clock cycles. Measurements obtained show that the instruction miss rate is 15% and the data miss rate is 6%, and that on average, 40% of all instructions are load/store types. The miss penalty for the cache is 20 cycles. What is the Average CPI?

Solution:

Average CPI = Base CPI + instruction miss cycles + data miss cycles

= 1 + 0.15 x 20 + 0.40 x 0.06 x 20 = 4.48 clock cycles

I-cache miss rate = 2%
D-cache miss rate = 4%
Miss penalty = 100 cycles
Base CPI (ideal cache) = 2
Load & stores are 36% of instructions



Miss cycles per instruction
I-cache: 0.02 × 100 = 2
D-cache: 0.36 × 0.04 × 100 = 1.44

Average CPI = Base CPI + instruction miss cycles + data miss cycles

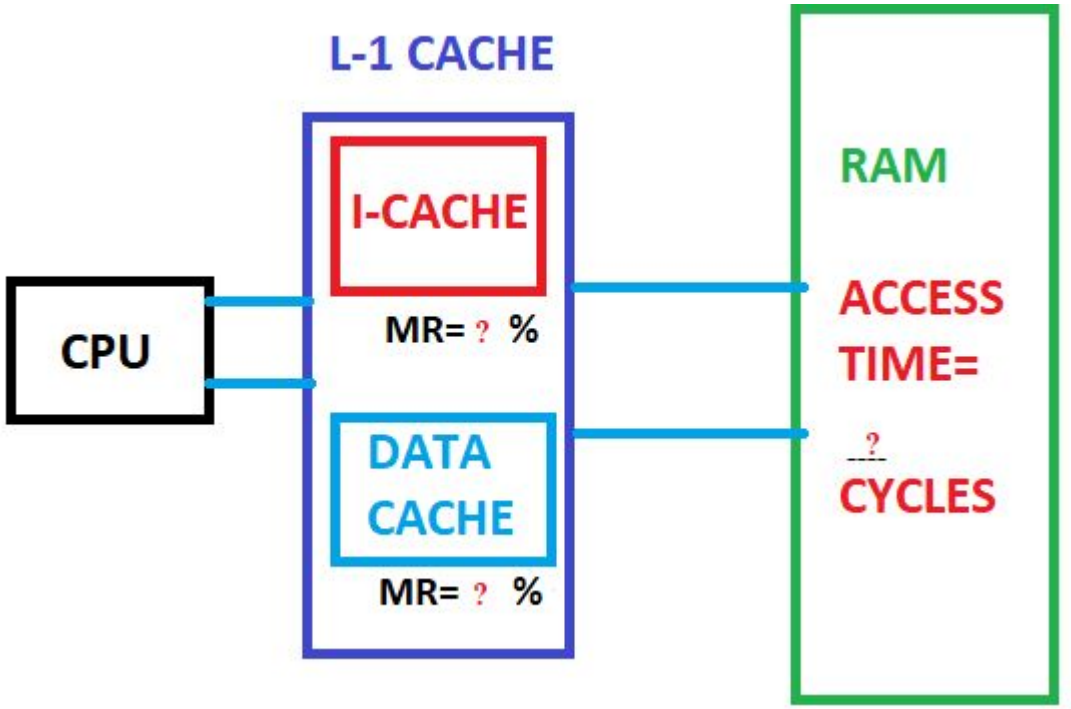
2 + 2 + 1.44 = 5.44

Speedup = Average CPI/Base CPI

= 5.44/2 =2.72

Ideal CPU is 2.72 times faster

- Assume
  - I-cache miss rate 3%.
  - D-cache miss rate 5%.
  - 40% of instructions reference data.
  - miss penalty of 50 cycles.
  - Base CPI is 2.
- What is the CPI including the misses?
- How much slower is the machine when account?



- Redo the above if the I-miss penalty is reduced to 10 (D-miss still 50)
- With I-miss penalty back to 50, what is performance if CPU (and the caches) are 100 times faster

Assume

- 5% I-cache misses.
- 10% D-cache misses.
- 1/3 of the instructions access data.
- The Base CPI is 4 clock cycles

What is the Average CPI if the miss penalty is 12 clock cycles?

What is the CPI if we upgrade to a double speed cpu+cache (base CPI =2 clock cycles), but use a slower memory having access time of 24 clock cycles?

- Assume that:
  - Instruction miss rate %2
  - Data miss rate %4
  - CPI is 2 (without any memory stalls)
  - Miss penalty 40 cycles
  - %36 of instructions are load/store
- Determine how much faster a machine would run with a perfect cache that never missed.

Instruction miss cycles = I x 0.02 x 40 = 0.80 I (**I** is # of instructions)

Data miss cycles = I x 0.36 x 0.04 x 40 = 0.58 I

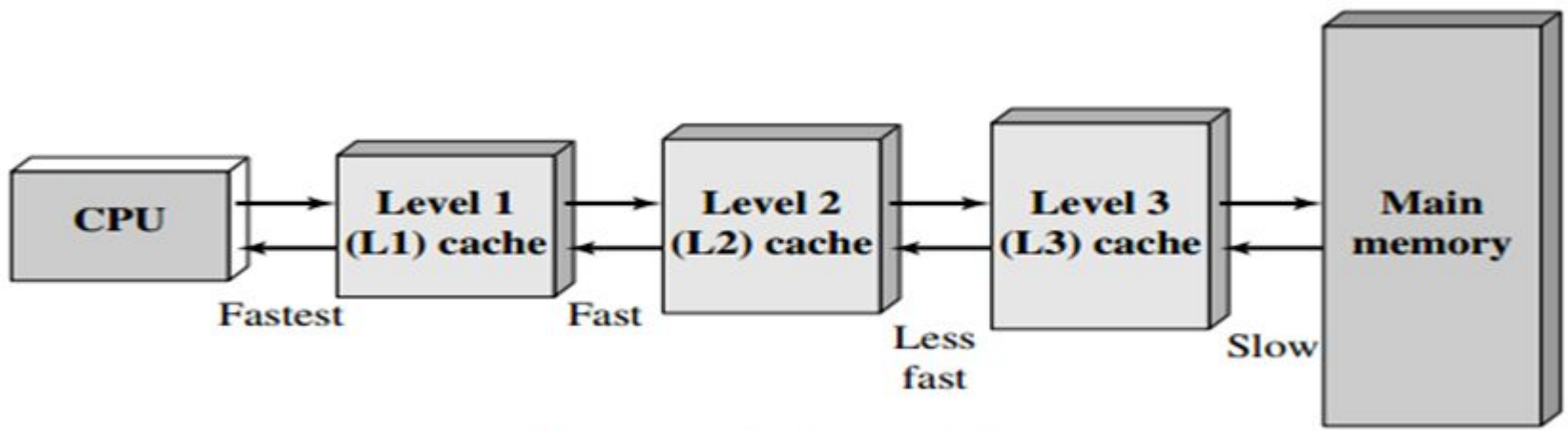
Total memory stall cycles = 0.80 I + 0.58 I = 1.38 I

$CPI_{stall} = 2 + 1.38 = 3.38$

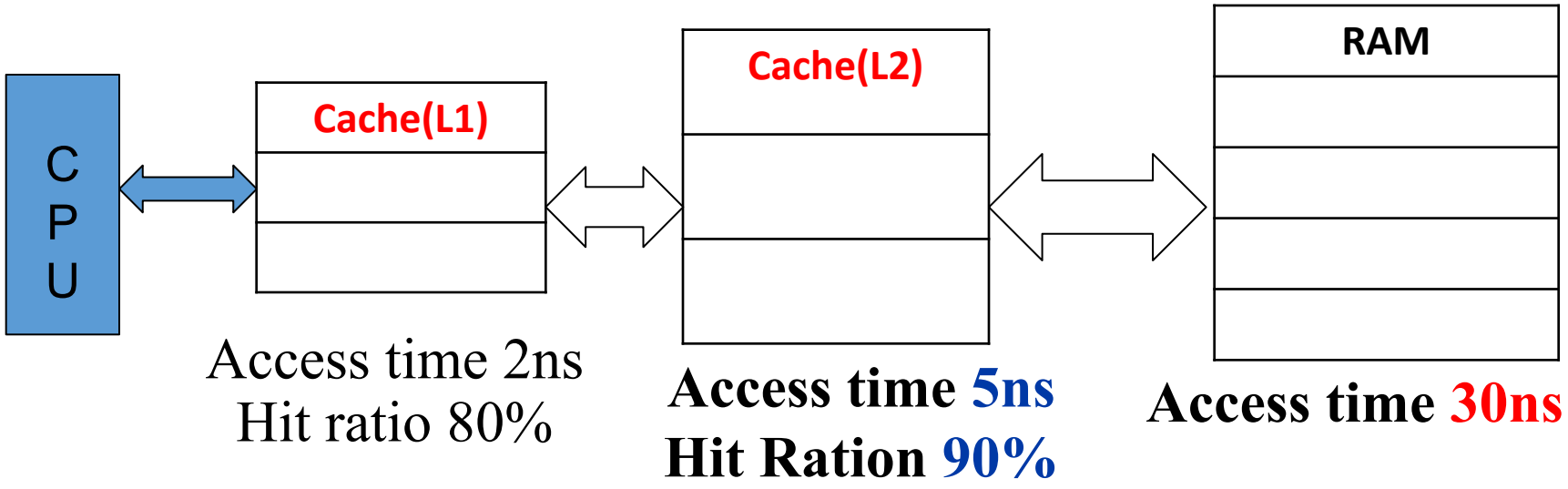
$$\frac{\text{CPU time with stalls}}{\text{CPU time with perfect cache}} = \frac{I \times CPI_{stall} \times \text{Clock cycle}}{I \times CPI_{perfect} \times \text{Clock cycle}} = \frac{3.38}{2} = 1.69$$

Multilevel Caches

- **Primary cache attached to CPU**
  - Small, but fast
- **Level-2 cache services misses from primary cache**
  - Larger, slower, but still faster than main memory
- **Main memory services L-2 cache misses**
- **Some high-end systems include L-3 cache**



Three-level cache organization



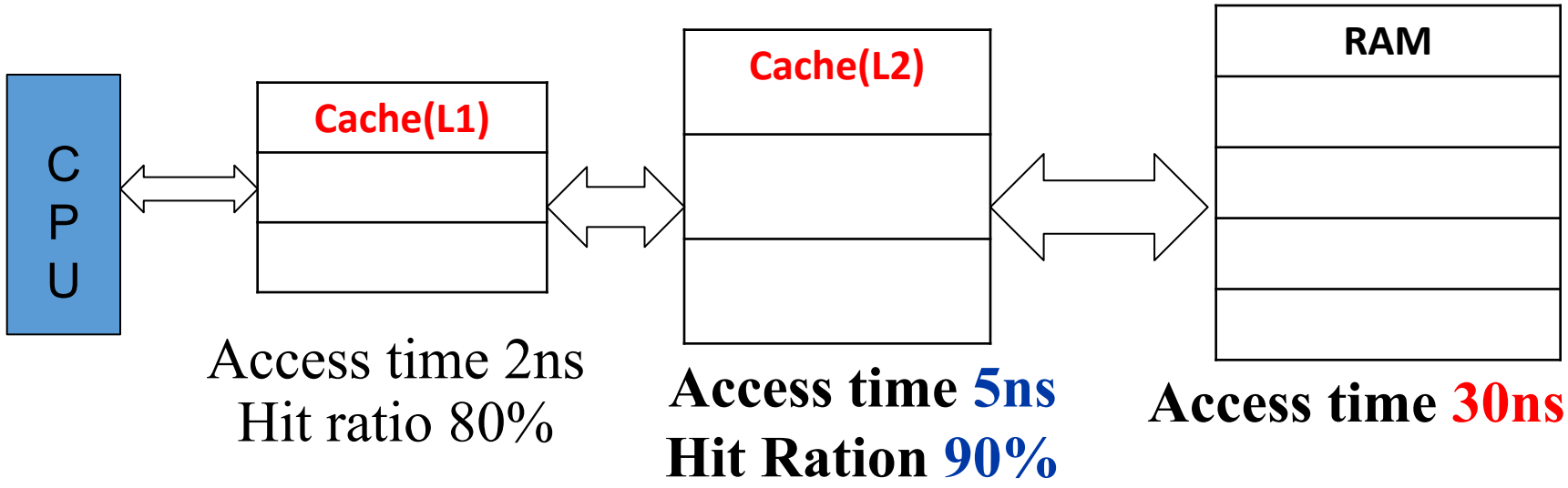
Suppose a program, initially loaded into RAM. The Hit ratios of L1 and L2 Caches are 80% and 90% respectively. Calculate average access time.

Solution: Access time from L1 cache: 0.8×2ns = 1.6ns

Access time from L2 cache: (1-0.8)×0.9×(2+5)ns = 1.26ns

Access time from RAM: [1-0.8-{(1-0.8)×0.9}]×(2+5+30)ns = 0.74ns

Average access time: 1.6ns + 1.26ns + 0.74ns = 3.6ns



Suppose a program, initially loaded into RAM, contains 1500 instructions. The Hit ratios of L1 and L2 Caches are 80% and 90% respectively. Calculate average access time. Also calculate total access time.

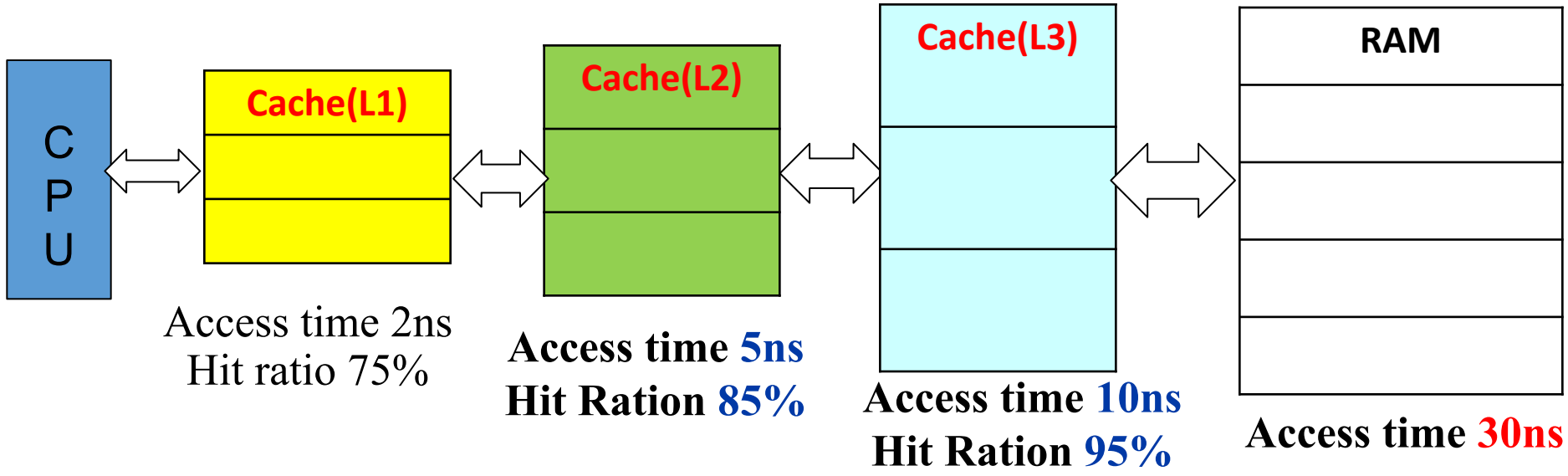
Solution: Access time from L1 cache: $0.8 \times 2ns = 1.6ns$

Access time from L2 cache: $(1-0.8) \times 0.9 \times (2+5)ns = 1.26ns$

Access time from RAM: $[1-0.8-\{(1-0.8) \times 0.9\}] \times (2+5+30)ns = 0.74ns$

Average access time: 1.6ns + 1.26ns + 0.74ns = 3.6ns

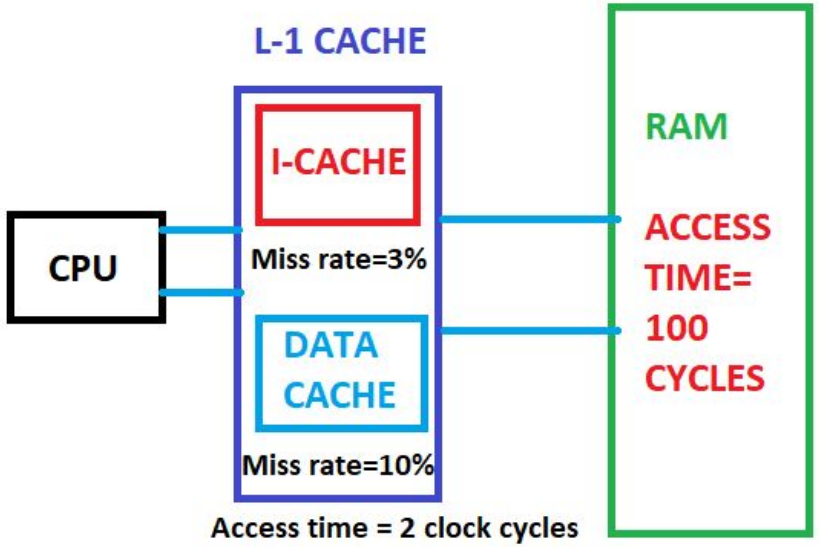
Total access time: $3.6ns \times 1500 = 5400ns$



Suppose a program, initially loaded into RAM, contains 1500 instructions. The Hit ratios of L1, L2 and L3 Caches are 75%, 85% and 95% respectively. Calculate average access time. Also calculate total access time.

Solution:

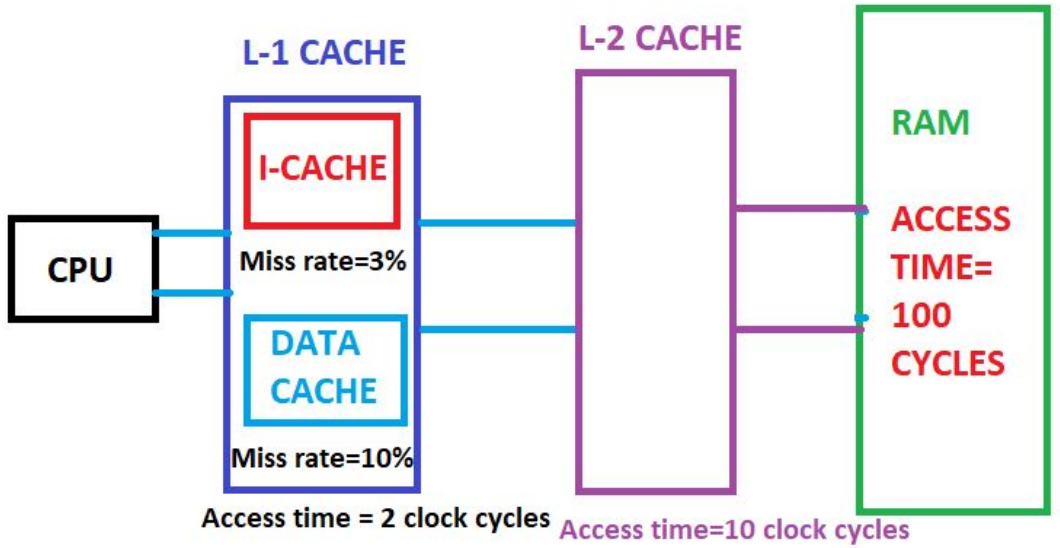
Suppose our processor has separate L1 instruction cache and data cache. Our CPI-base is 2 clock cycles, whereas memory accesses take 100 cycles. Our Instruction cache miss rate is 3% while our Data cache miss rate is 10%. 40% of our instructions are loads or stores.



a. What is our processor's Average CPI?

Average CPI = CPI base + L1 inst miss cycles + L1 data miss cycles
= 2 + 1 ×0.03 ×100 + 0.4×0.1×100
= 2 +0.07 × 100 = 9 cycles

To improve the performance our processor, we add a unified L2 cache between the L1 caches and memory. Our L2 cache has a hit time of 10 cycles and a global miss rate of 2%.



Here

Global miss rate = miss rate of L2 cache

b. What is our new Average CPI?

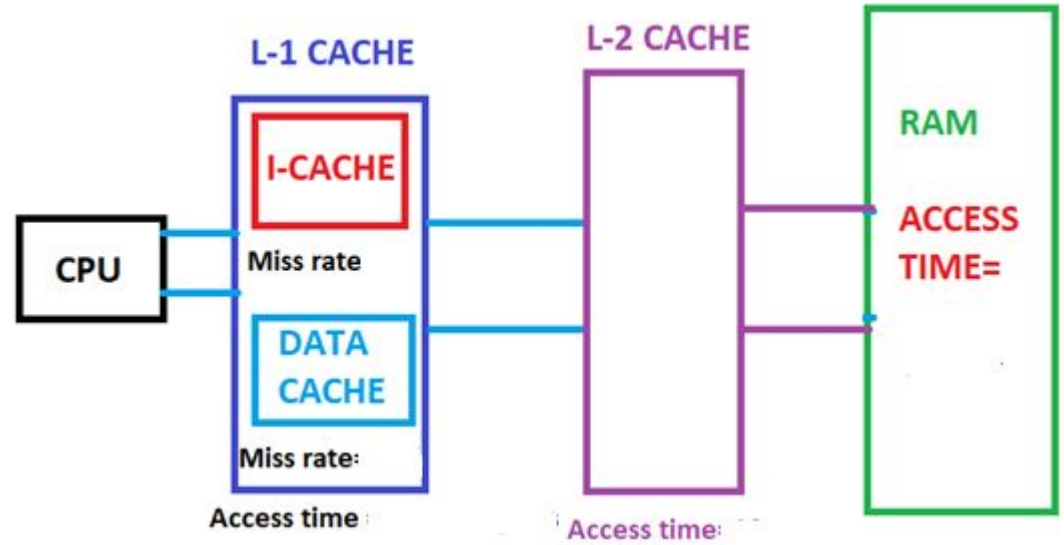
new Average CPI = CPI base + L1 inst miss cycles + L1 data miss cycles + L2 inst miss cycles + L2 data miss cycles

= 2 + (1 ×0.03 × 10) + (0.4 ×0.1 × 10) + (1×0.02 × 100) + (0.4 ×0.02 × 100)

= 2 + 0.3 + 0.4 + 2 + 0.8

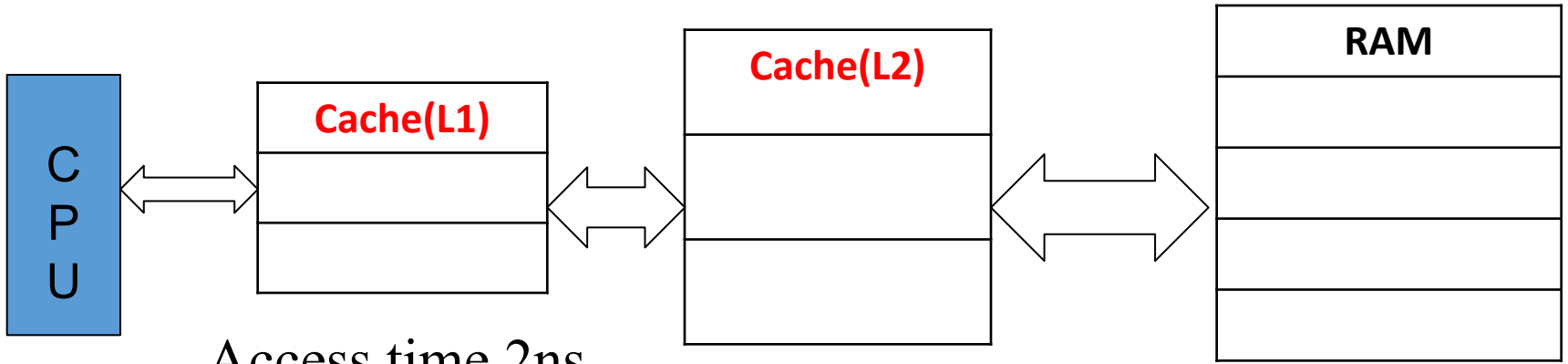
= 2 + 3.5 = 5.5 cycles

Multilevel Cache AMAT



- AMAT = L1 HT + L1 MR × L1 MP
  - Now L1 MP depends on other cache levels
- L1 MP = L2 HT + L2 MR × L2 MP
  - If more levels, then continue this chain (i.e. $MP_i = HT_{i+1} + MR_{i+1} \times MP_{i+1}$)
  - Final MP is main memory access time
- For two levels:

AMAT = L1 HT + L1 MR × (L2 HT + L2 MR × L2 MP)



Suppose a program, initially loaded into RAM, contains 1500 instructions. The Hit ratios of L1 and L2 Caches are 80% and 90% respectively. Calculate average access time.

$$AMAT = L1\ HT + L1\ MR \times L1\ MP$$

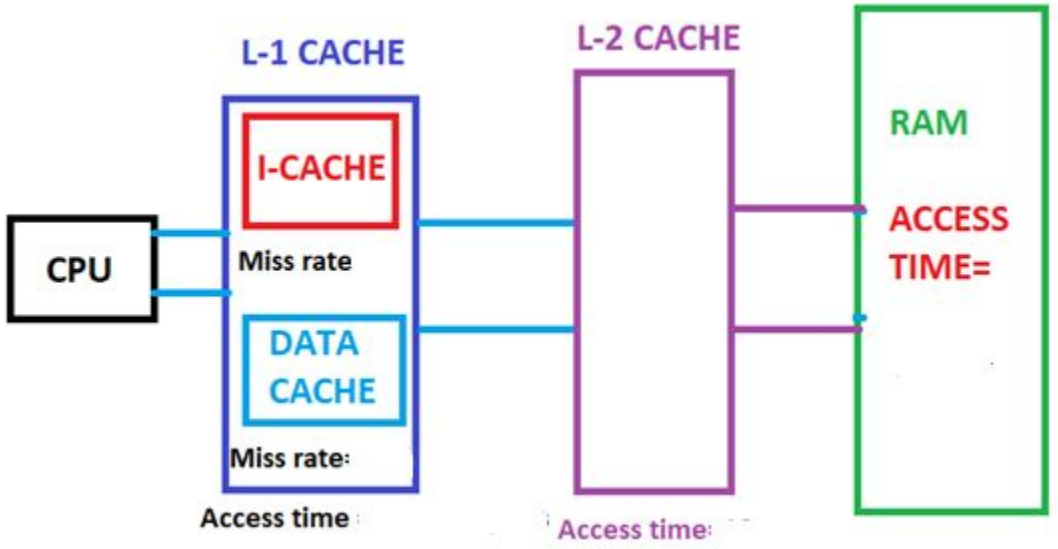
$$L1\ MP = L2\ HT + L2\ MR \times L2\ MP$$

L1HT = 2ns, L1MR = 0.2, L2HT = 5ns, L2MR = 0.1

AMAT = L1HT + L1MR (L2 HT + L2MR x L2MP) = 2ns + 0.2 (5 + 0.1x30)

= 2ns + 0.2 (5 + 0.1 x 30) ns = 2 + 0.2 x 8 = 2 + 1.6 = 3.6 ns

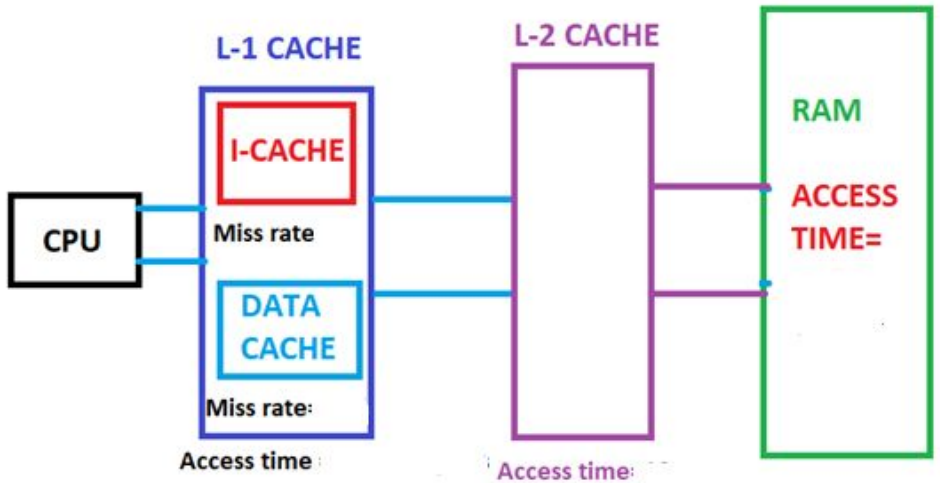
- Assume
  - L1 I-cache miss rate 4%
  - L1 D-cache miss rate 5%
  - 40% of instructions reference data
  - L2 miss rate 6%
  - L2 time of 15ns
  - Memory access time 100ns
  - Base CPI of 2
  - Clock rate 400MHz



- How many instructions per second does this machine execute?
- How many instructions per second would this machine execute if the L2 cache were eliminated?
- How many instructions per second would this machine execute if both caches were eliminated?
- How many instructions per second would this machine execute if the L2 cache had a 0% miss rate (L1 as originally specified)?
- How many instructions per second would this machine execute if both L1 caches had a 0% miss rate?

Global Miss Rates

fraction of total accesses that miss at L1 and L2



For a two level cache, we know:

$$MR_{global} = L1\ MR \times L2\ MR$$

AMAT:

- $AMAT = L1\ HT + L1\ MR \times (L2\ HT + L2\ MR \times L2\ MP)$

$$= L1\ HT + L1\ MR \times L2\ HT + MR_{global} \times L2\ MP$$

Cache Performance Example

- Given
  - I-cache miss rate = 2%
  - D-cache miss rate = 4%
  - Miss penalty = 100 cycles
  - Base CPI (ideal cache) = 2
  - Load & stores are 36% of instructions

- Miss cycles per instruction
  - I-cache: 0.02 × 100 = 2
  - D-cache: 0.36 × 0.04 × 100 = 1.44
- Actual CPI = 2 + 2 + 1.44 = 5.44
  - Ideal CPU is 5.44/2 =2.72 times faster

Average Access Time

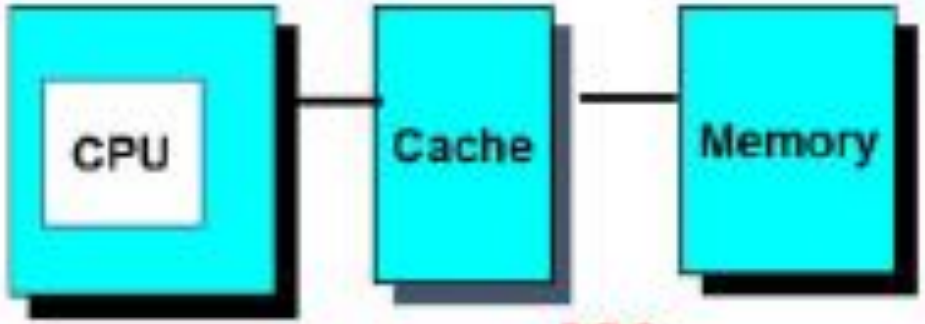
- Hit time is also important for performance
- Average memory access time (AMAT)
  - AMAT = Hit time + Miss rate × Miss penalty
- Example
  - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - AMAT = 1 + 0.05 × 20 = 2ns
    - 2 cycles per instruction

Multilevel Cache Example

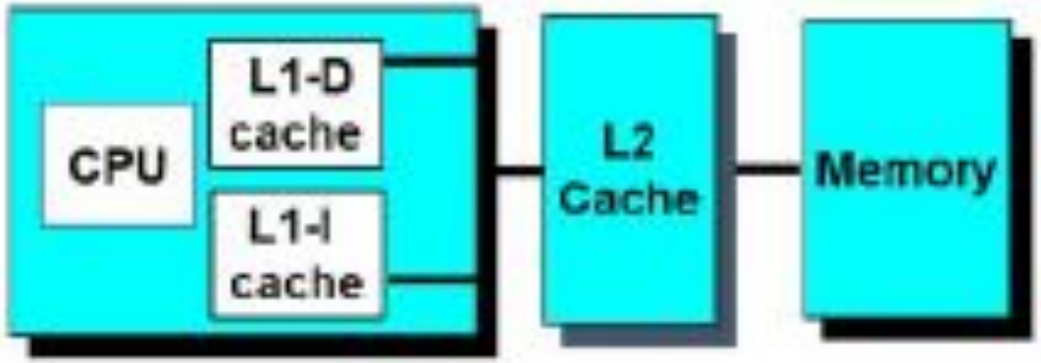
- Given
  - CPU base CPI = 1, clock rate = 4GHz
  - Miss rate/instruction = 2%
  - Main memory access time = 100ns
- With just primary cache
  - Miss penalty = 100ns/0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = 9

Example (cont.)

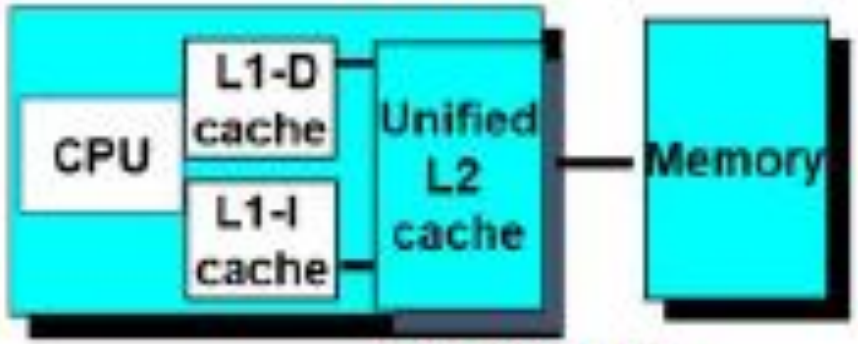
- Now add L-2 cache
  - Access time = 5ns
  - Global miss rate to main memory = 0.5%
- Primary miss with L-2 hit
  - Penalty = 5ns/0.25ns = 20 cycles
- Primary miss with L-2 miss
  - Extra penalty = 500 cycles
- CPI = 1 + 0.02 × 20 + 0.005 × 400 = 3.4
- Performance ratio = 9/3.4 = 2.6



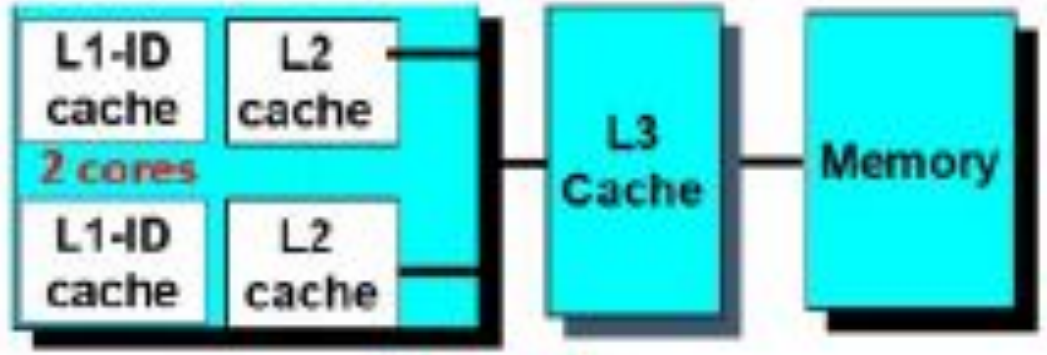
386



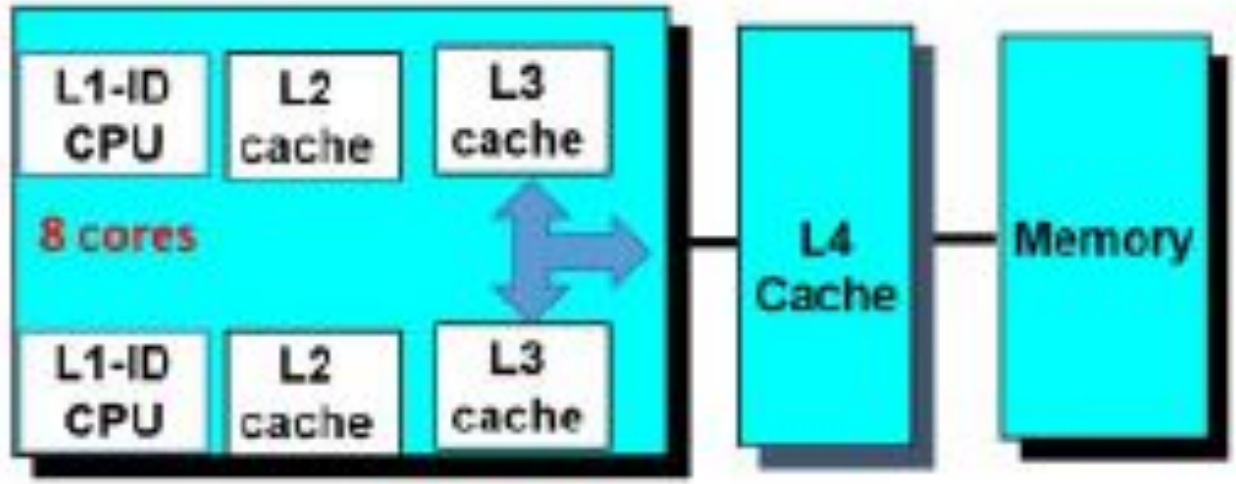
Pentium



Pentium 4



Power6



Power 8