Advance Computer Architecture and x86 ISA

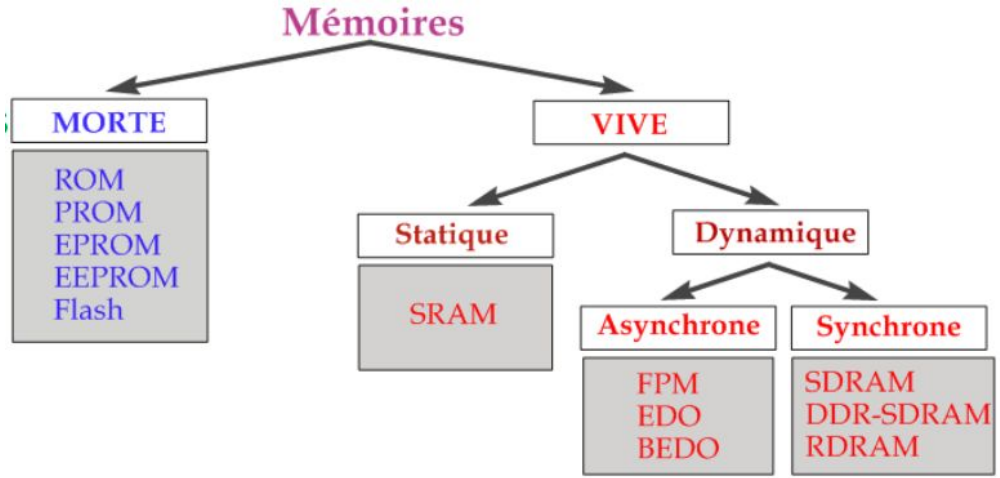
University of Science and Technology of Hanoi

MS. LE Nhu Chu Hiep

Introduction

Recall

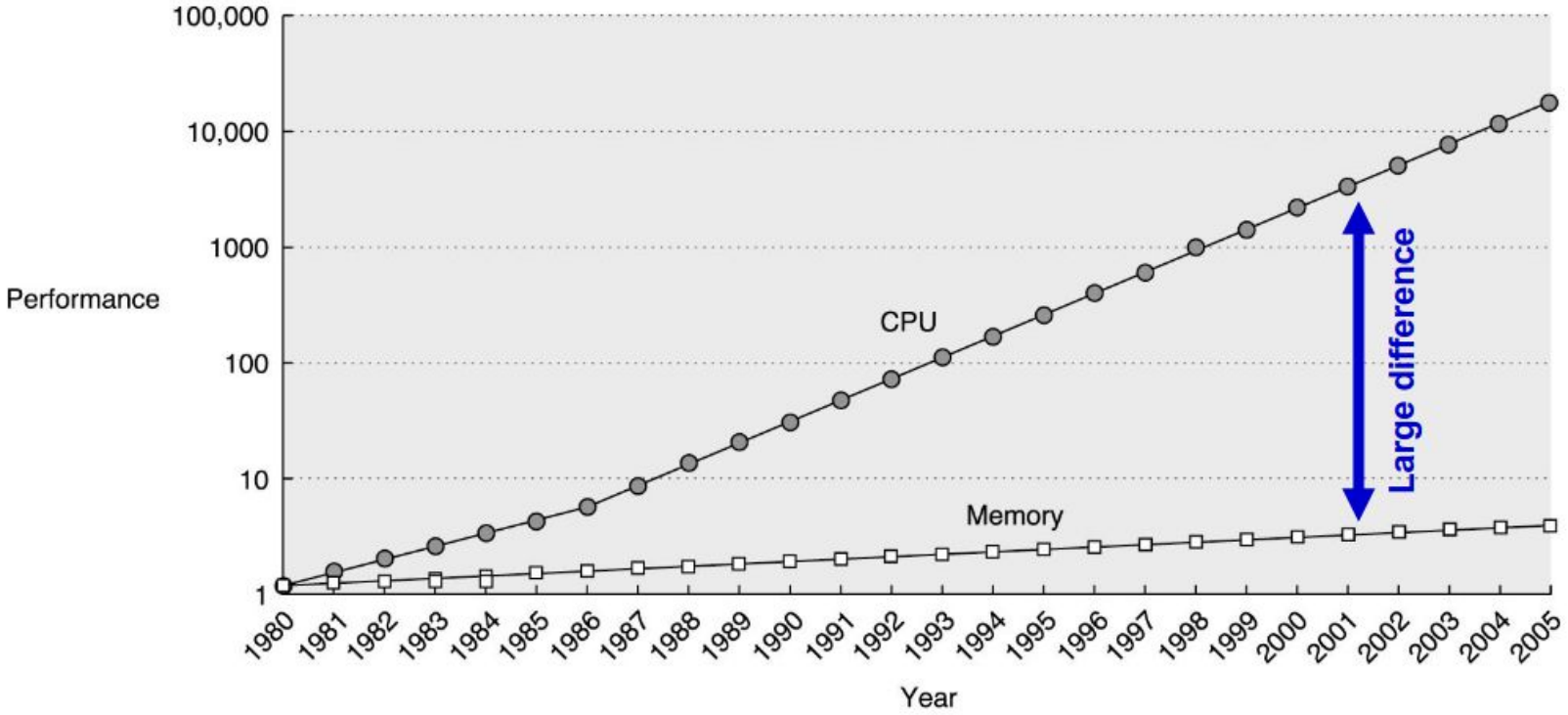
- The memories
  - Store program data and instructions
- Memory types
  - Permanent memory
  - Temporary memory
- Permanent memory
  - Nonvolatile, large, cheap, slow
- Temporary memory
  - Volatile, small, expensive, fast



Why defining a memory organization ?

- Application complexity increase
  - Data to store dramatically increases
  - Size of memories increases
- Evaluation of computer
  - 1980, computer has just few kilobytes of memory
  - Today, several gigabyte of memory are needed
- The memories must be
  - Very **large** in size
  - But also need to propose **small** access time
  - **Contradiction !!!**

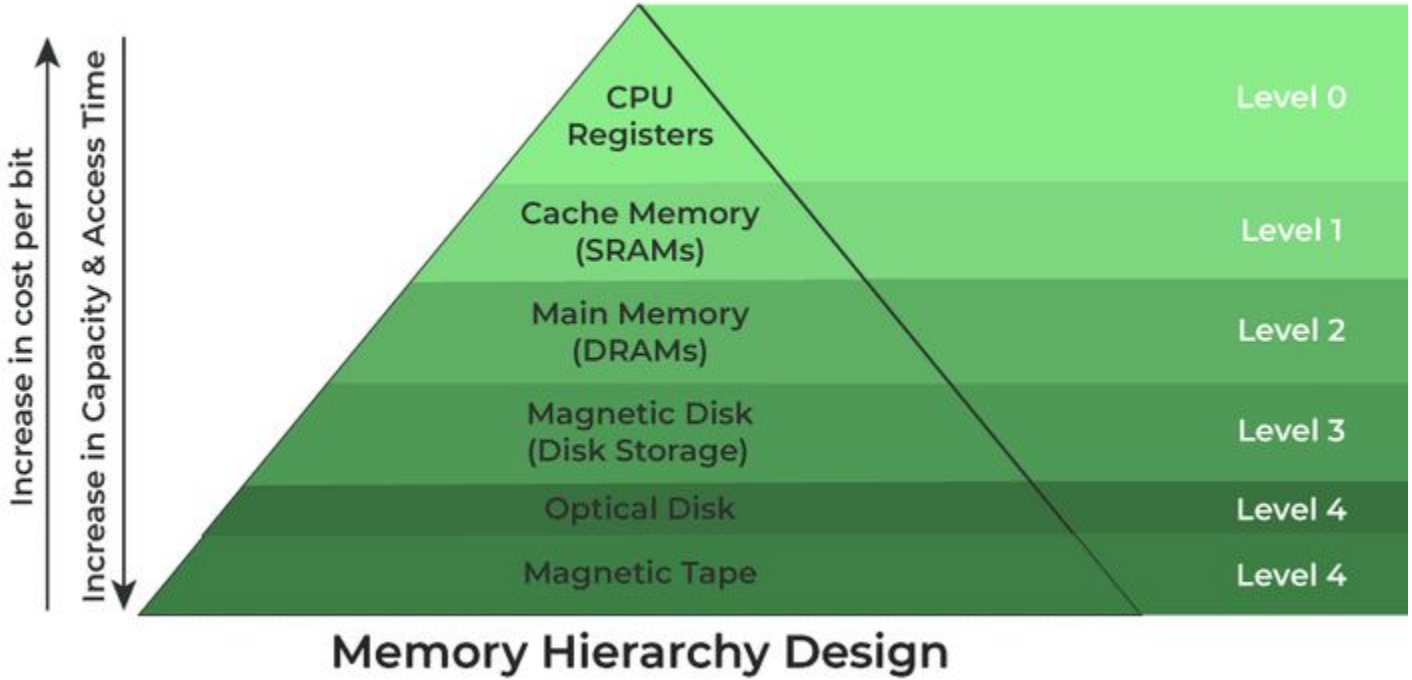
Evaluation of processors and memory performance



Example of access time

Technologies	Access time	Human time	Capacities	Price $ / Mo
Register	1 à 2 ns	1 s	64 * 64 bits	Include in the Processor
Internal cache	3 ns	3 s	32 ko	Include in the Processor
External cache	25 ns	25 s	4 Mo	40
Main Memory	200 ns	3 min	1 Go	2,5
Drive	12 ms	139 jours	10 Go	0,010
Tape	10 à 20 s	315 ans et plus	50 Go	< 0,05

Level representation of memories

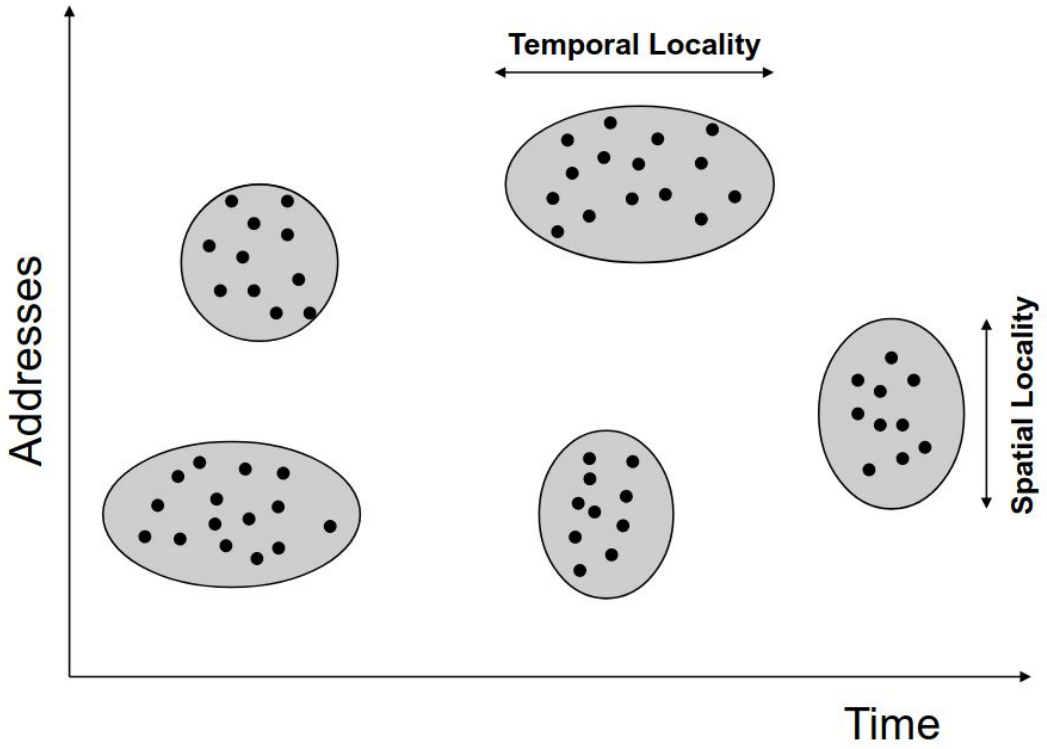


Localities

Principle of locality

- The tendency of a processor to access the same set of memory locations repetitively over a short period of time
- Spatial locality
  - If the processor execute instruction from address @i, it probably want to execute the instruction stored at address @i+1
- Temporal locality
  - If the processor accesses to data d at time t, he will probably reuse this data d in a short time

Localities of memory access



Example of localities

```
for (i = 0 ; i < N ; i++) {
    for (j = 0 j < N ; j++) {
        y[i] =y[i] + a[i][j] * x[j] ;
    }
}
```

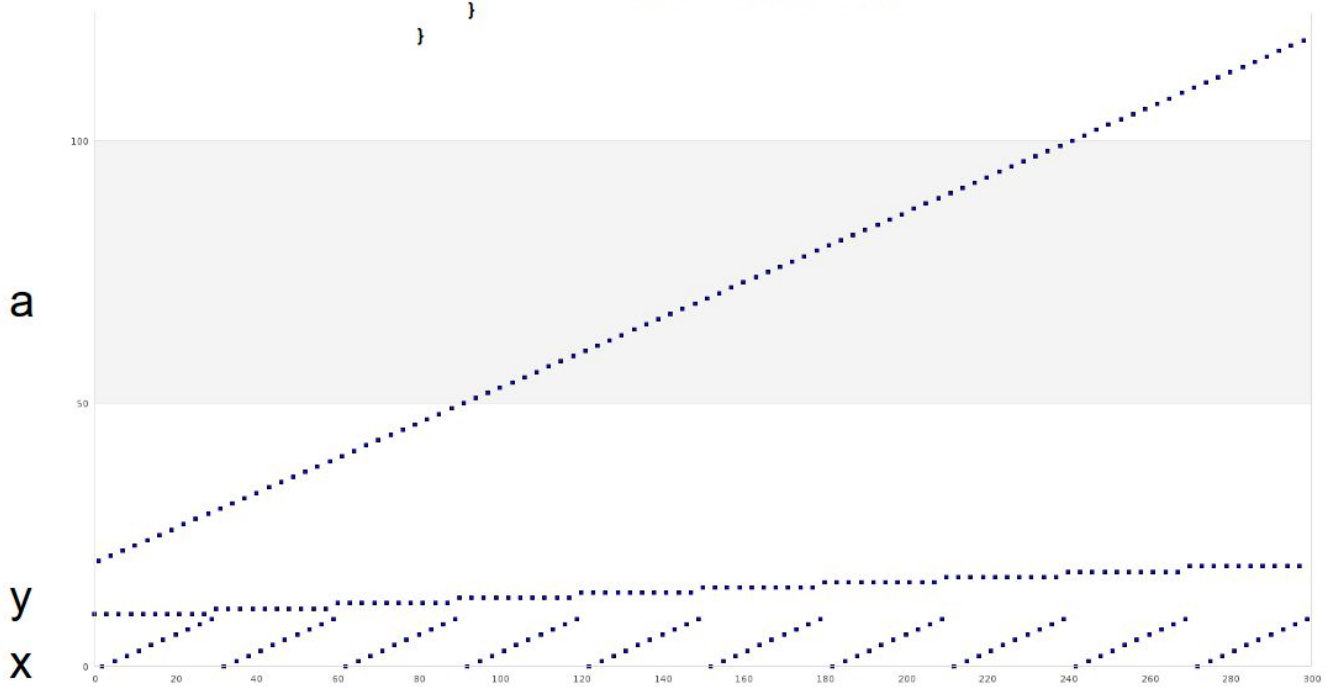
```
Boucle   LD      R0, R1
         ADD     R2, R3, R4
         MULT    R5, R2, R7
         SUB     R10, R10, #1
         BRnz    R10, Boucle
```

- y[i] : spatial and temporal localities
- a[i][j] : spatial locality
- x[j] : spatial and temporal localities

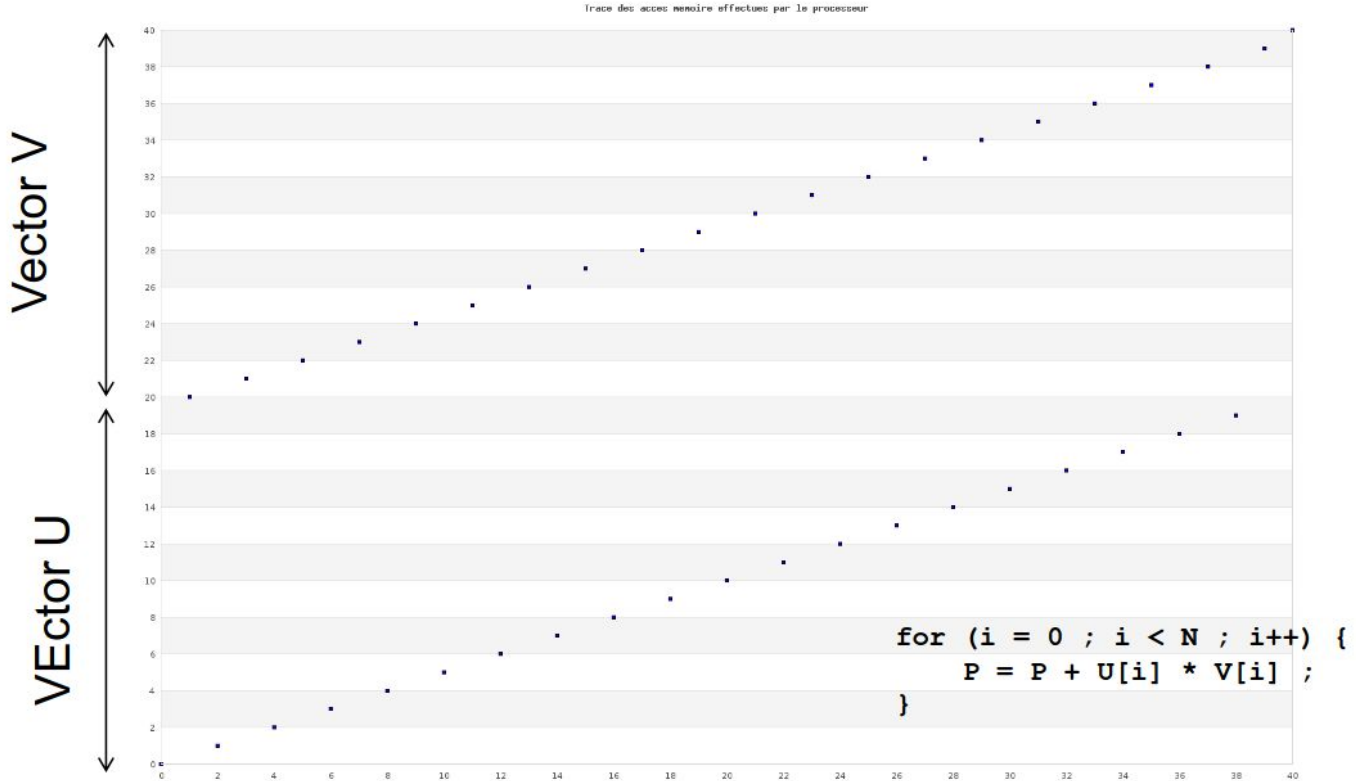
- In the loop, large reuse of instructions

Example of data access localities

```
for (i = 0 ; i < N ; i++) {
    for (j = 0 j < N ; j++) {
          y[i] =y[i] + a[i][j] * x[j] ;
    }
}
```



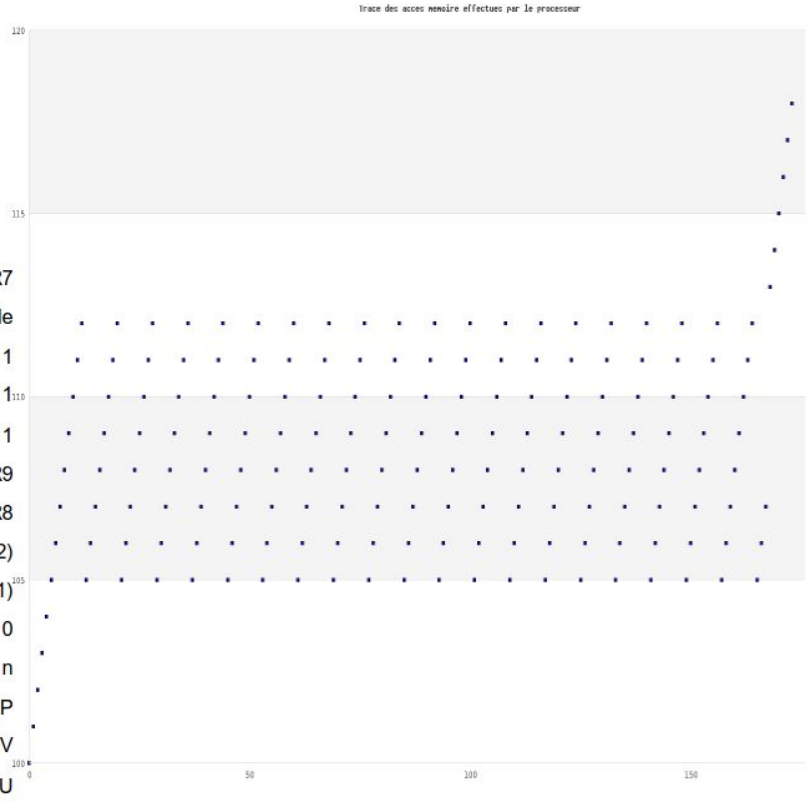
Example of data access localities for Scala product



Example of instruction access localities for Scala product

Load in memory

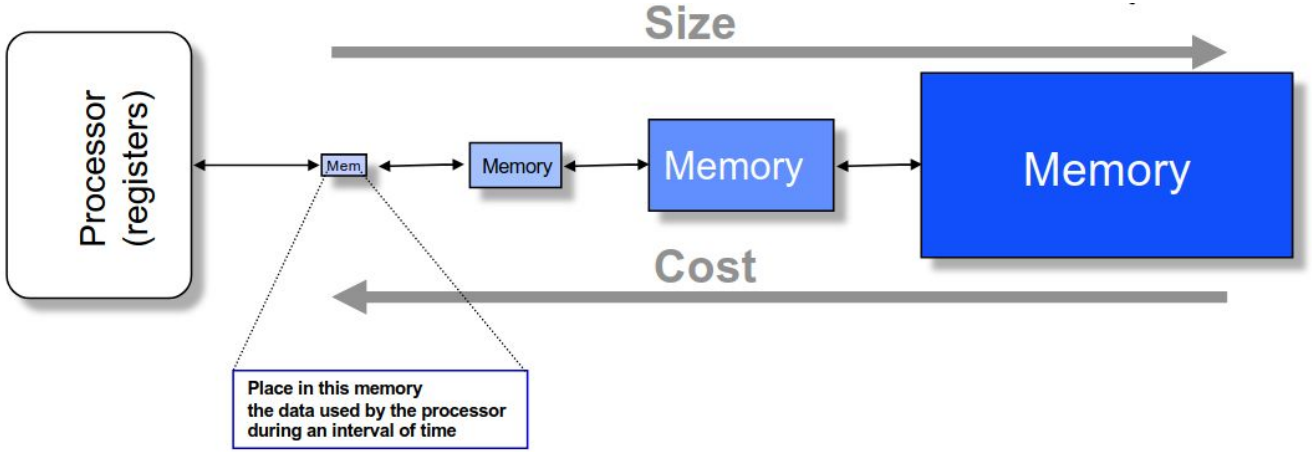
```
program 100
            li R1, U
            li R2, V
            li R3, P
            li R4, n
            li R7, 0
boucle
            li R8, (R1)
            li R9, (R2)
            mult R9, R9, R8
            add R7, R7, R9
            add R1, R1, 1
            add R2, R2, 1
            sub R4, R4, 1
            brnz R4, boucle
            si (R3), R7
            exit
end
```



Memory Organization

Idea

- Place a hierarchy of memory
  - Small memory near the processor, fast, but small
  - Large memory far the processor, large but slow



Performance of a memory hierarchy

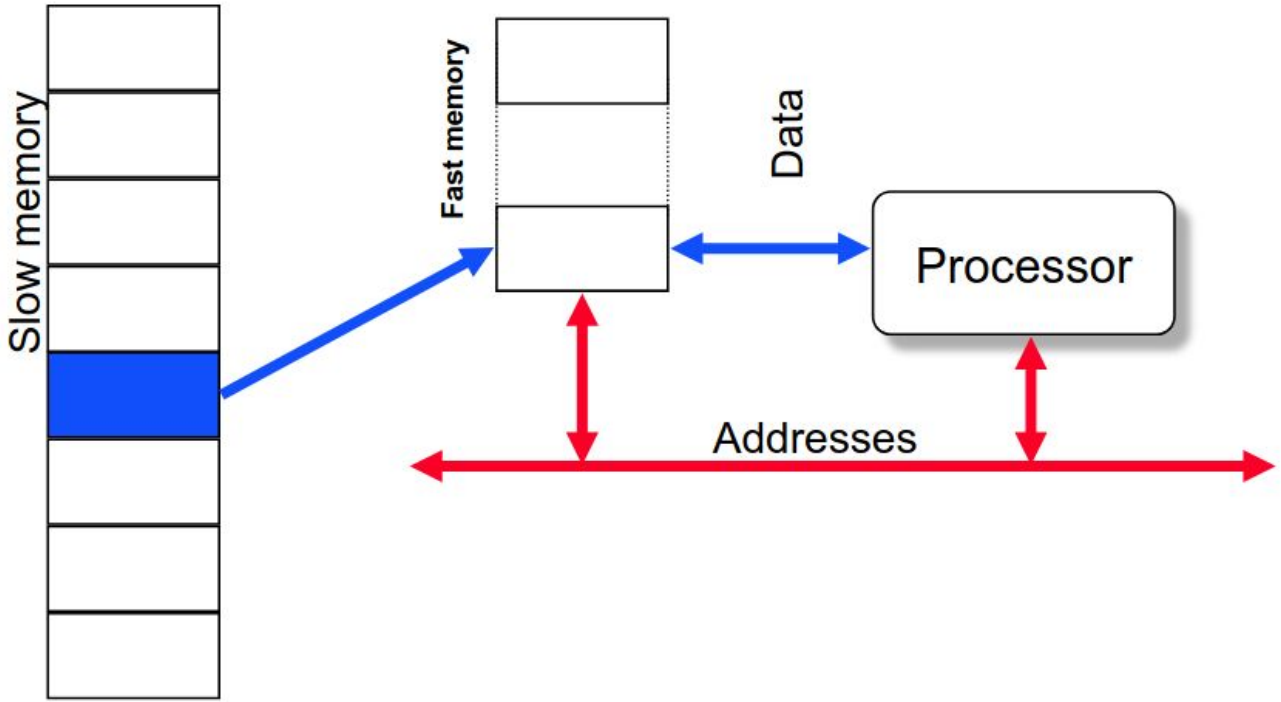
- Performance of a memory hierarchy

$$\text{Perf} = \frac{\text{Time with fast memory}}{\text{Time with the memory hierarchy}}$$

- Gain obtained with memory hierarchy

$$\text{Gain} = \frac{\text{Time with slow memory}}{\text{Time with the memory hierarchy}}$$

How it work ?



Performance and gain

Let N : the block size
P : the number of adat read or written in the block
Ts : access time for slow memory
Tf : access time for fast memory

$$\text{Perf} = \frac{P * Tf}{2 * N * Ts + P * Tf}$$

$$\text{Acc} = \frac{P * Ts}{2 * N * Ts + P * Tf}$$

Performance and gain (cont.)

- Condition to ensure for efficient memory hierarchy

$$2 * N * Ts + P * Tf < P * Ts$$

- Interest of memory hierarchy

$$P > \frac{2 * N * Ts}{Ts - Tf}$$

Interest of memory hierarchy

- The more the difference between memories is, the more the data must be reused
- If P is small, then the memory hierarchy is not so interesting
- Example:
  - Let N = 256, Ts = 2 * Tf

$$P > \frac{2 * 256 * 2 * Tf}{2 * Ts - Tf}$$

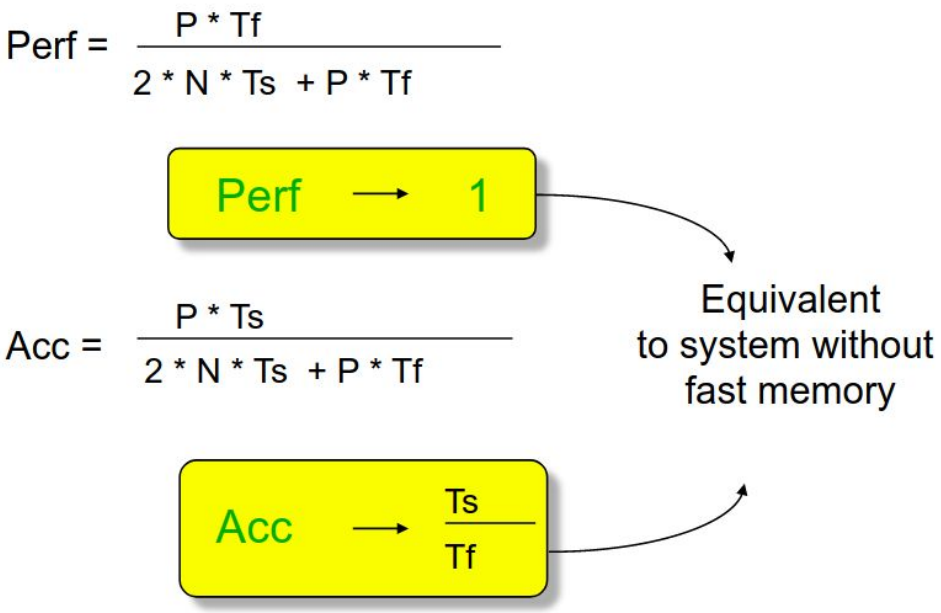
$$P > 2 * 256 * 2$$

$$P > 4 * 256$$

In average, each data must be read/write 4 times

Interest of memory hierarchy (cont.)

- If P -> infinity then:



Thank you for you listening