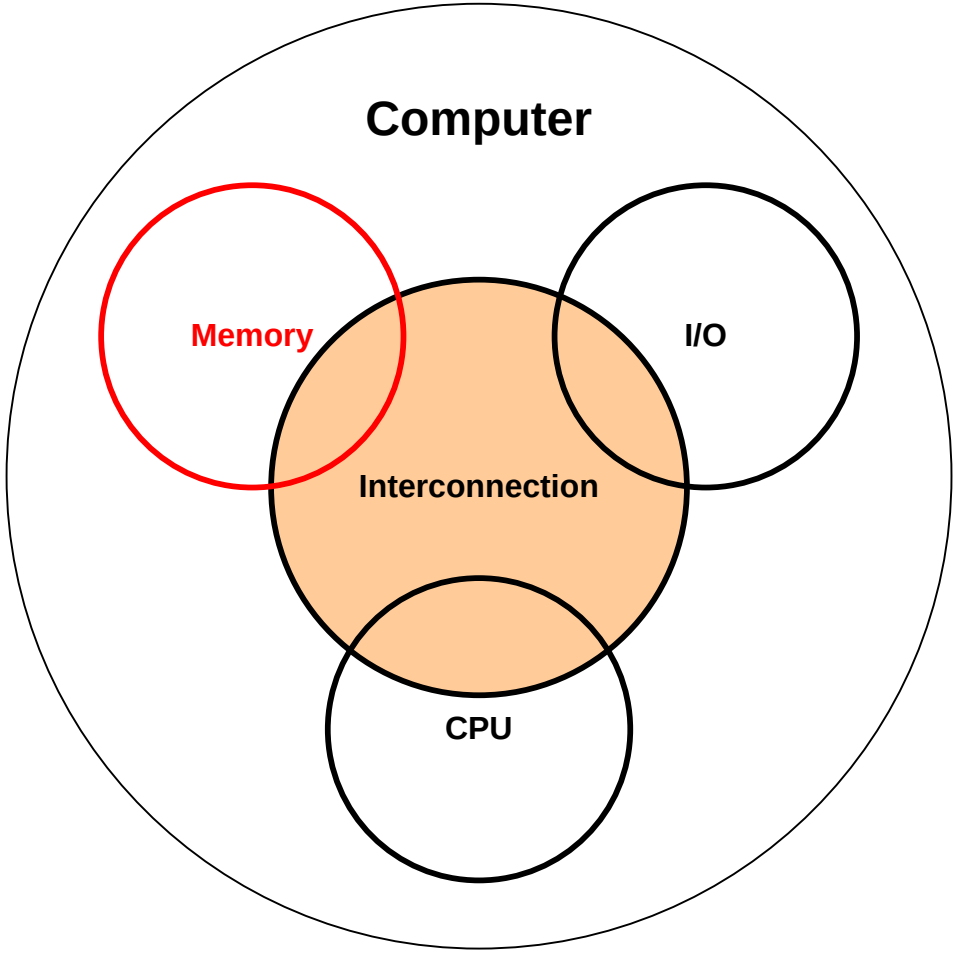
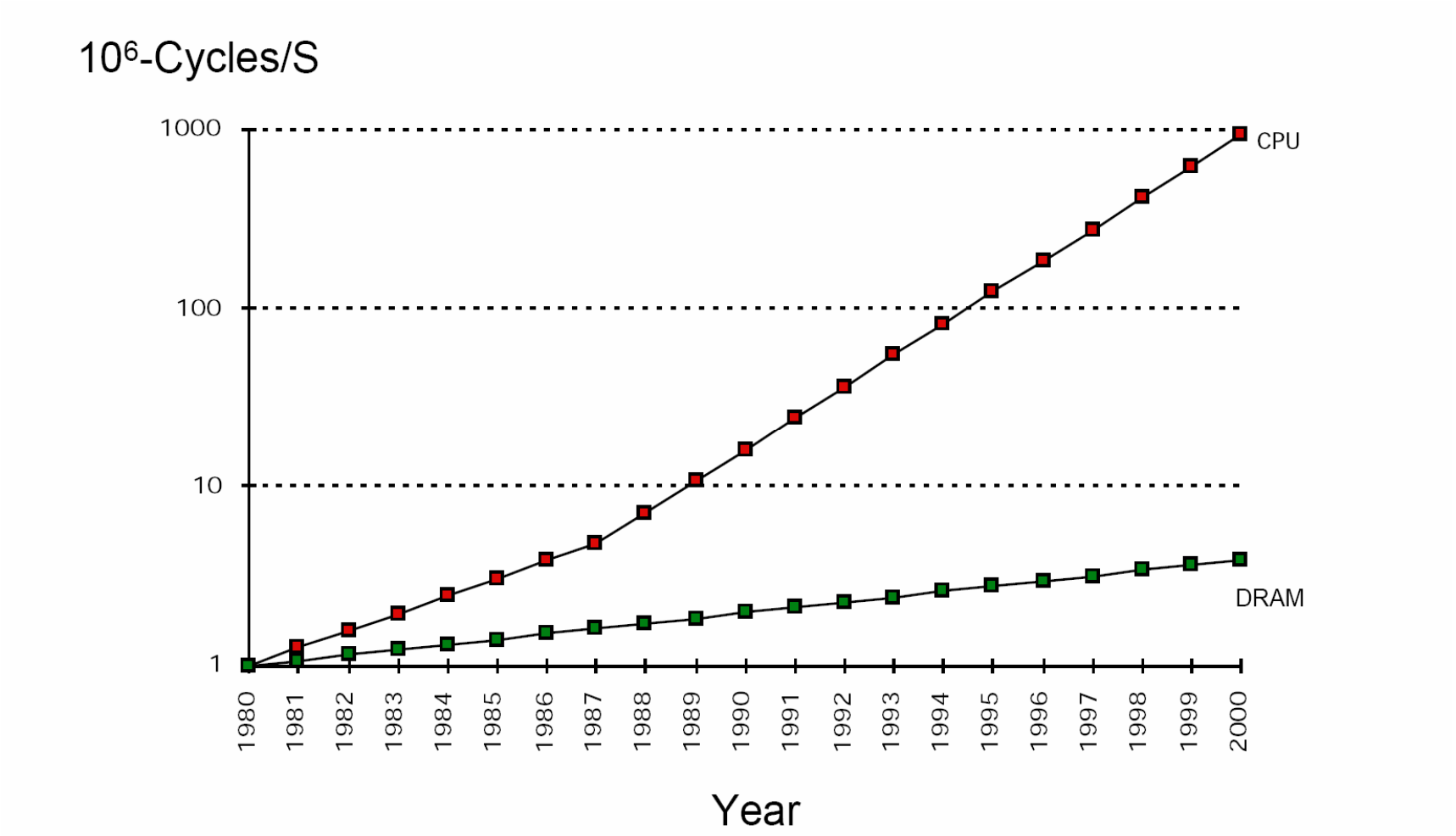
Memory



The CPU / Memory Gap (1)



The CPU / Memory Gap (2)

To illustrate the problem consider "typical" delays, measured in ns.

- **Clock Period: 0.3ns**
- **Instructions : 1-4 instructions/clock (4-way super-scalar)**
- **On-chip small-fast SRAM (Level-1 cache): 0.3-0.6ns (1-2 clocks).**
- **On-chip large-fast SRAM (Level-2 cache) 4-6ns (12-18 clocks).**
- **Off-chip large-fast SRAM (Level-3 cache) 7-14ns (20-40 clocks)**
- **Off chip large-slow DRAM (Main memory) 90-120ns (270-360 clocks)**

Question: How often does the computer access memory?

Key Idea

Keep the most often-used data in a small, fast SRAM (often local to CPU chip)

Refer to Main Memory only rarely, for remaining data.

The reason this strategy works:

LOCALITY

The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)
- Last 15 years, HW relied on localilty for speed

Typical Memory Reference Patterns

MEMORY TRACE

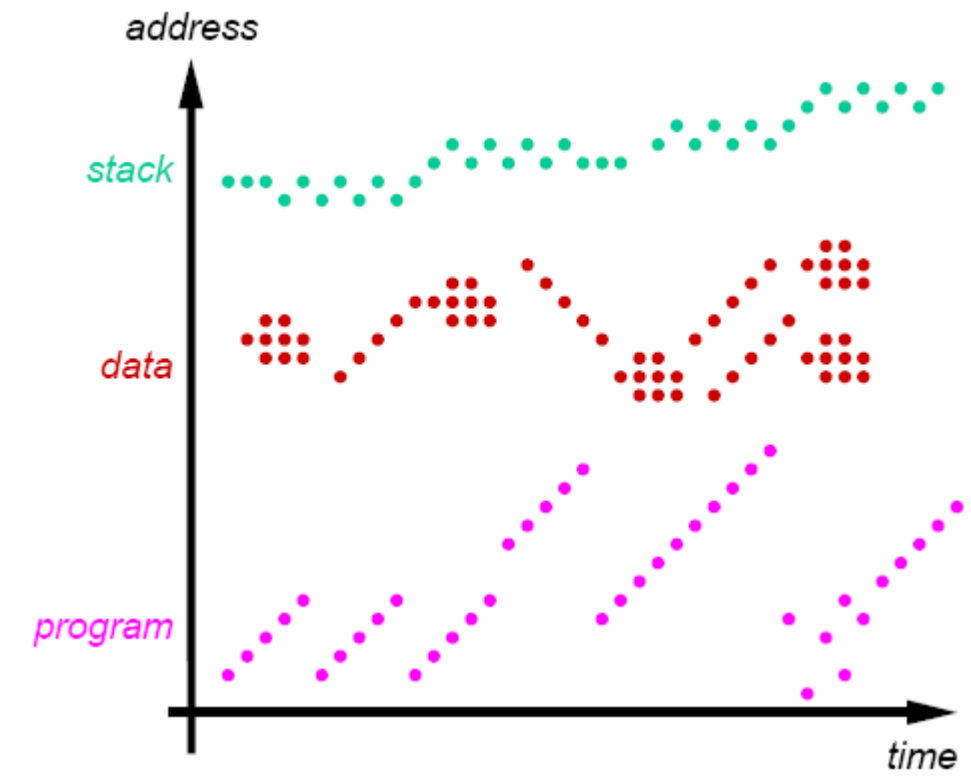
A temporal sequence of memory references (addresses) from a real program.

TEMPORAL LOCALITY

If an item is referenced, it will tend to be referenced again soon

SPATIAL LOCALITY

If an item is referenced, nearby items will tend to be referenced soon.



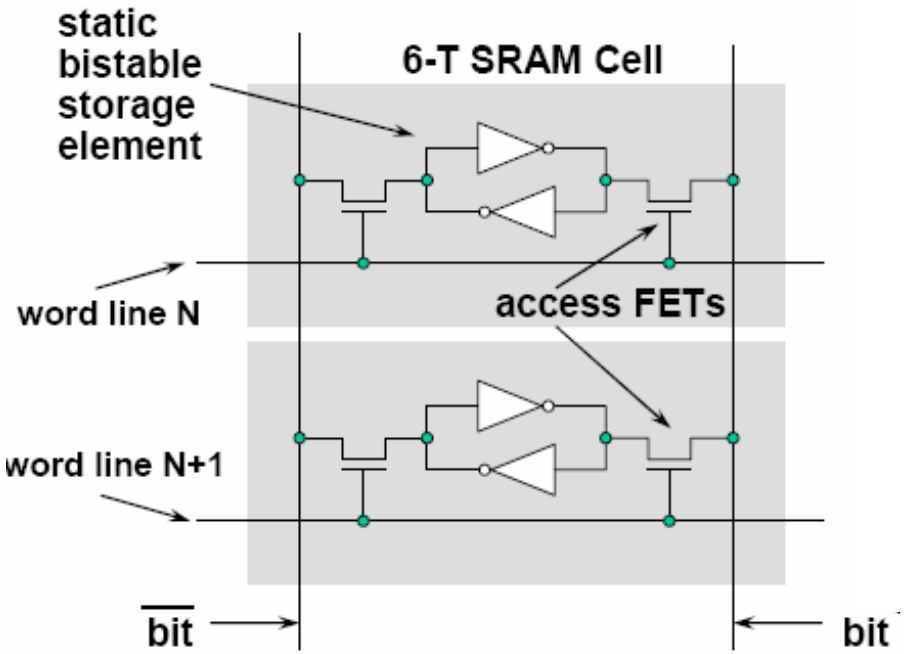
Memory Technology

(a short reminder)

Semi-conductor Memories

Type	Category	Erasing	Writing	Volatile
RAM	Read/Write	Elec. Byte	electrical	yes
ROM	Read only	impossible	mask	no
PROM	Read only	impossible	electrical	no
EPROM	mostly Read	UV, chip	electrical	no
Flash	mostly Read	elect, block	electrical	no
EEPROM	mostly Read	elect, byte	electrical	no

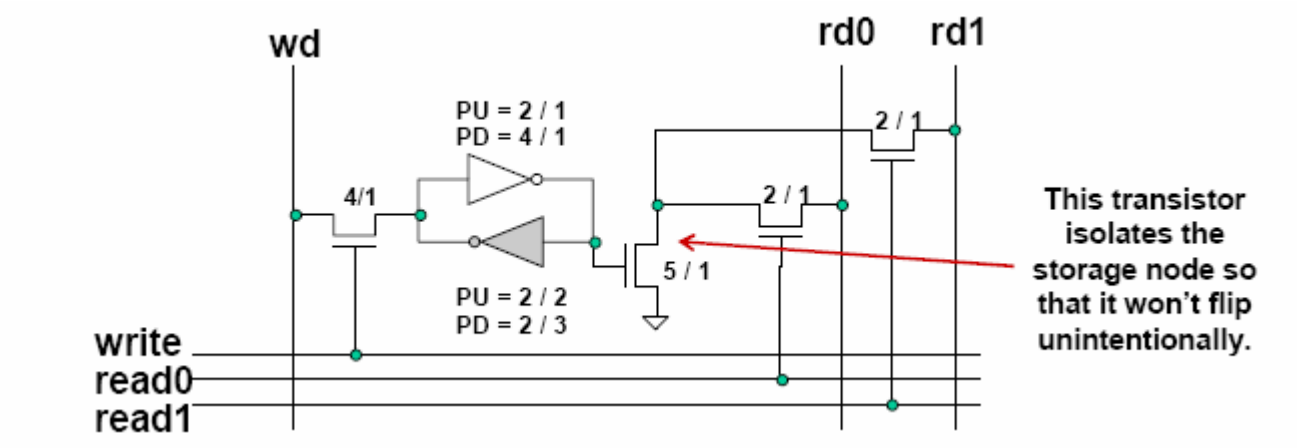
SRAM Memory Cell



- There are two bit-lines per column, one supplies the bit the other it's complement.
- On a Read Cycle
  - A single word line is activated (driven to "1"), and the access transistors enable the selected cells, and their complements, onto the bit lines.
- Writes are similar to reads,
  - except the bit-lines are driven with the desired value of the cell.
- The writing has to "overpower" the original contents of the memory cell.

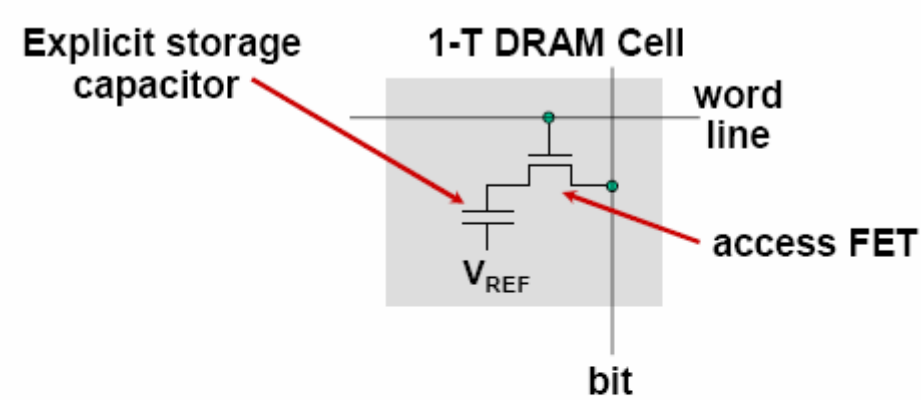
Multiport SRAM (a.k.a. Register Files)

One can increase the number of SRAM ports by adding access transistors. By carefully sizing the inverter pair, so that one is strong and the other weak, we can assure that our WRITE bus will only fight with the weaker one, and the READs are driven by the stronger one. Thus minimizing both access and write times.

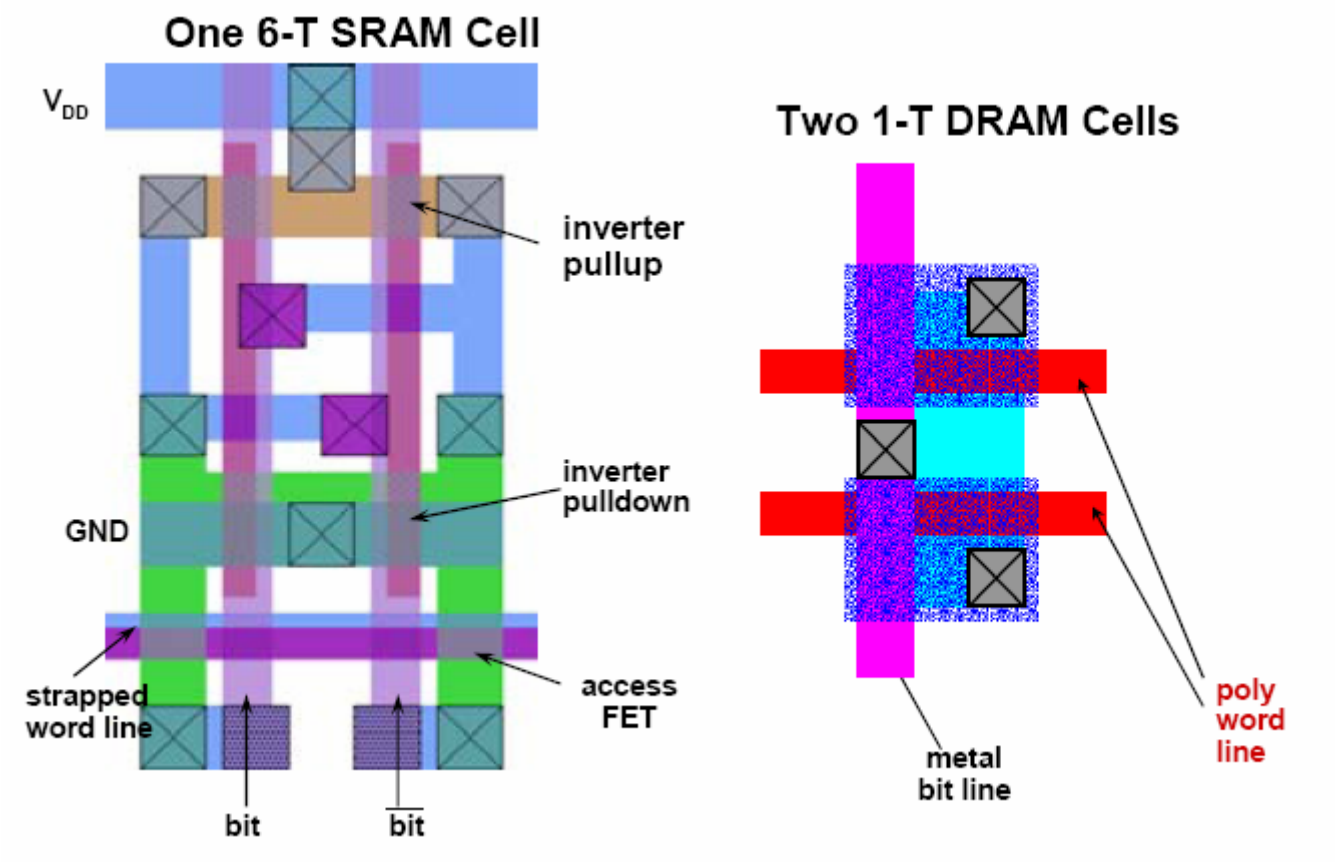


1-T Dynamic RAM

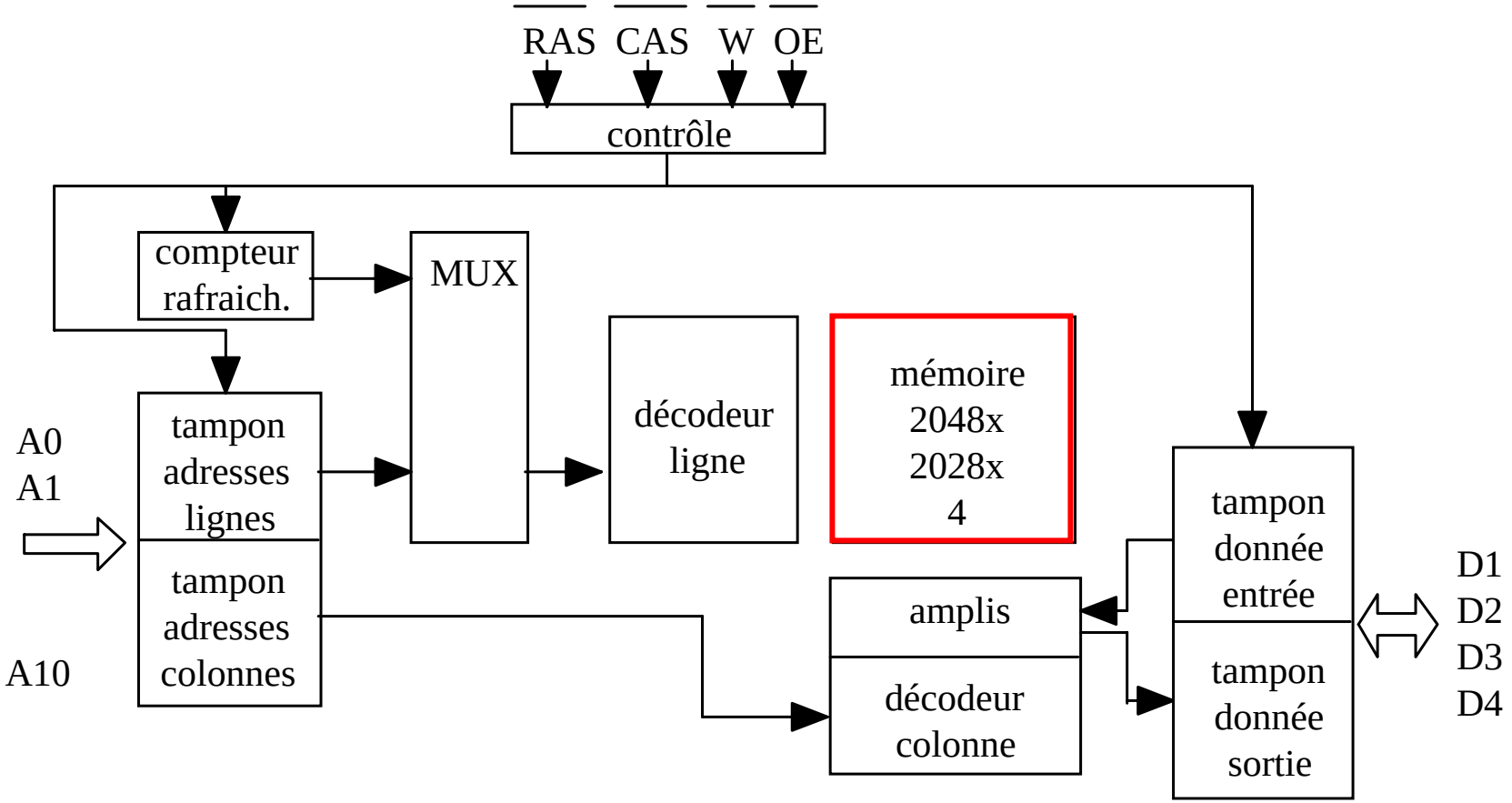
Six transistors/cell may not sound like much, but they can add up quickly. What is the fewest number of transistors that can be used to store a bit?



Side-by-side Comparison



Typical DRAM



DRAM Fundamentals

DRAM access is made of two parts:

- **Row Access: Select a bank and a row and read a whole row into the sense amplifiers. (Open a DRAM page)**
- **Column Access: Select an open page and read one word out.**

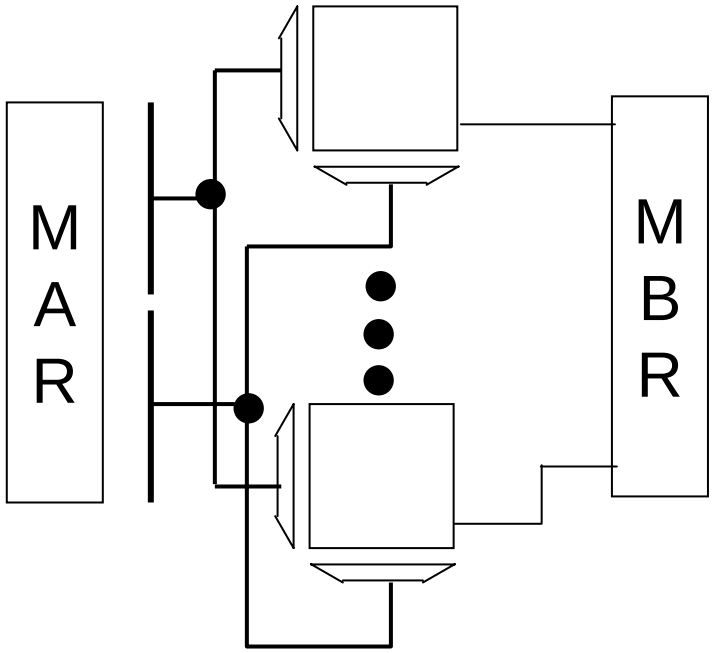
Multiple internal banks => one can keep multiple open pages.

Typically multiple column accesses are needed to get a whole cache line out.

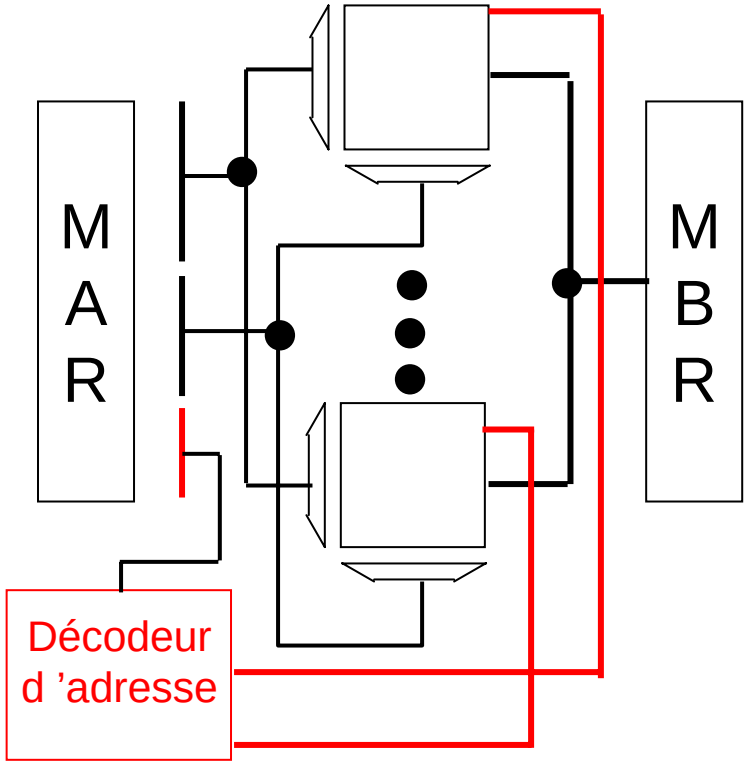
Column accesses are pipelined

- **It takes about 25ns-30ns from the start of the column access command until the data come out.**
- **A sequence of column access commands can be issued, one every clock cycle (one every ~7.5ns).**

Memory Organization

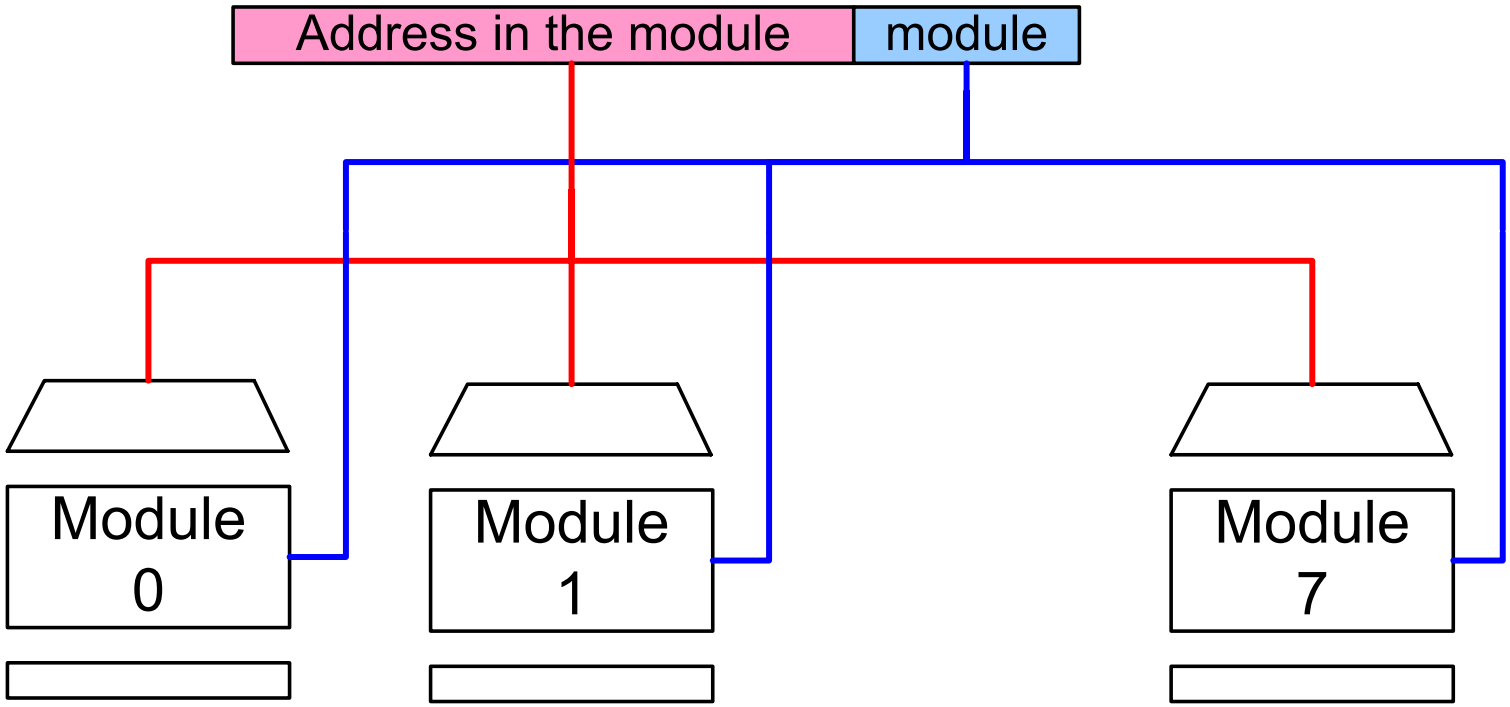


Wider memory word

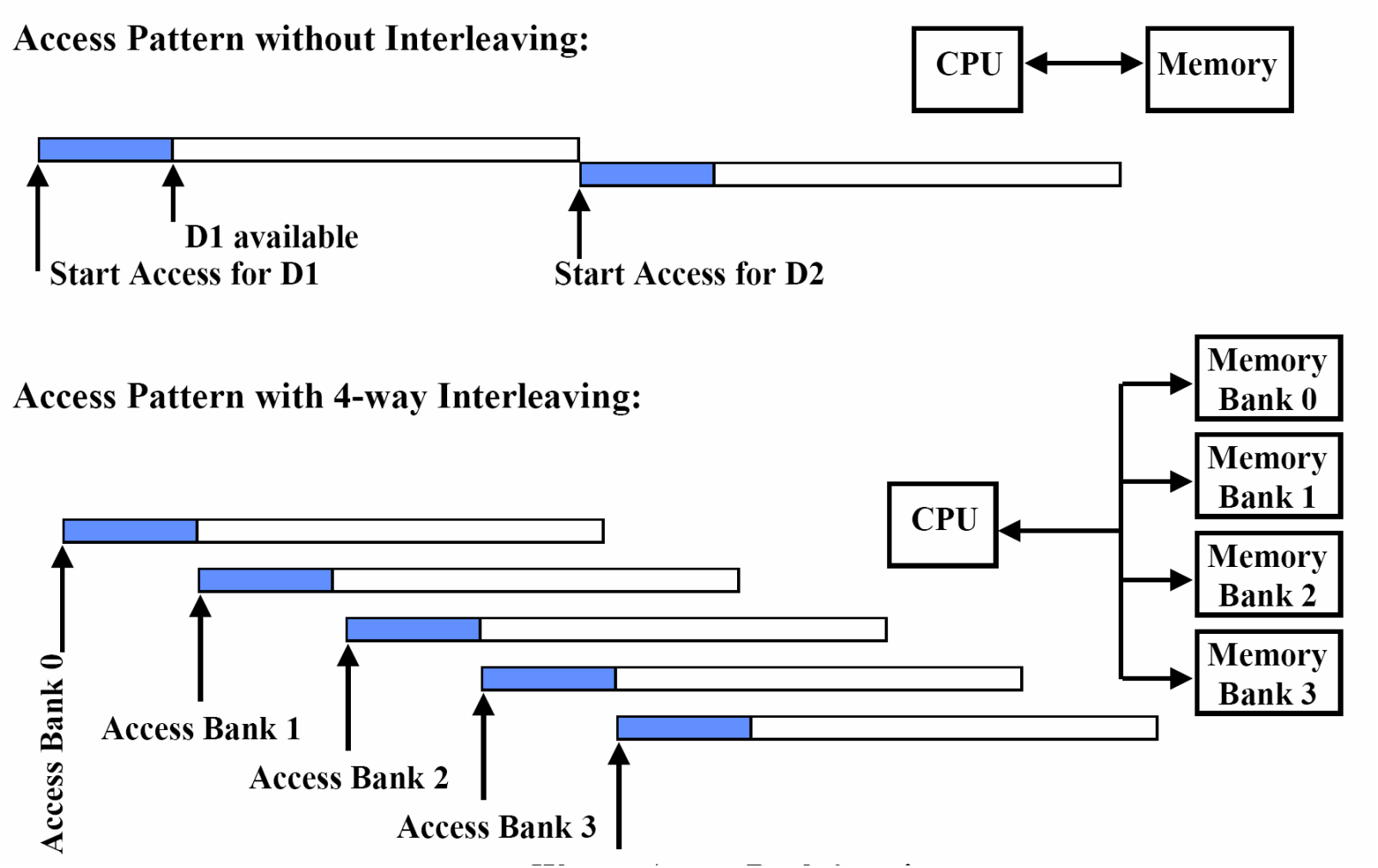


Larger memory space

Interleaved Memories



Interleaving: Increasing Bandwidth



Summary of Memory Technology

DRAM is slow but cheap and dense:

- **Good choice for presenting the user with a BIG memory system**
- **Uses one transistor, must be refreshed.**

SRAM is fast but expensive and not very dense:

- **Good choice for providing the user FAST access time.**
- **Uses six transistors, holds state as long as power is supplied.**

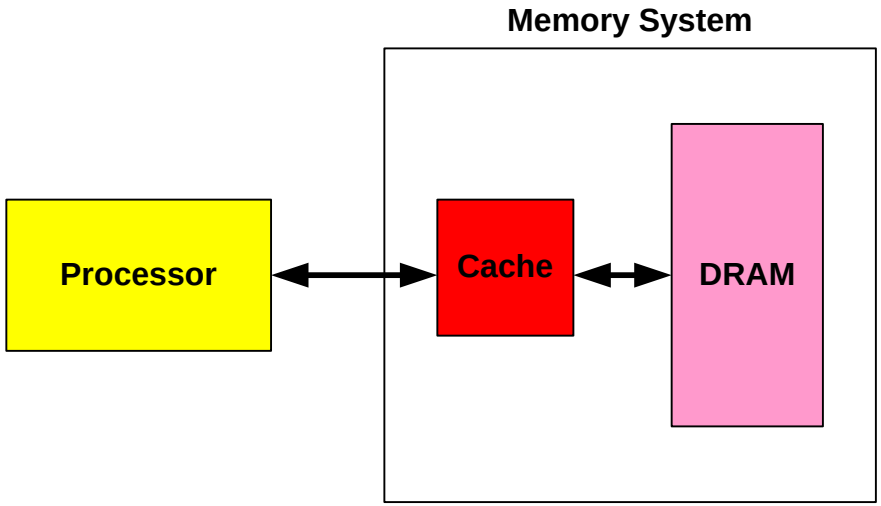
GOAL:

- **Present the user with large amounts of memory using the cheapest technology.**
- **Provide access at the speed offered by the fastest technology.**

Next: Caches

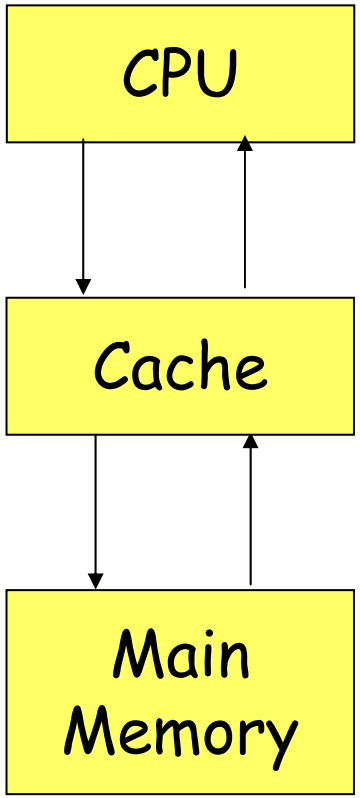
Cache

Motivation for Caches



- **Motivation:**
  - **Large memories (DRAM) are slow**
  - **Small memories (SRAM) are fast**
- **Make the *average access time* shorter by:**
  - **Servicing most accesses from a small, fast memory.**
- **Reduce the *bandwidth* required of the large memory.**

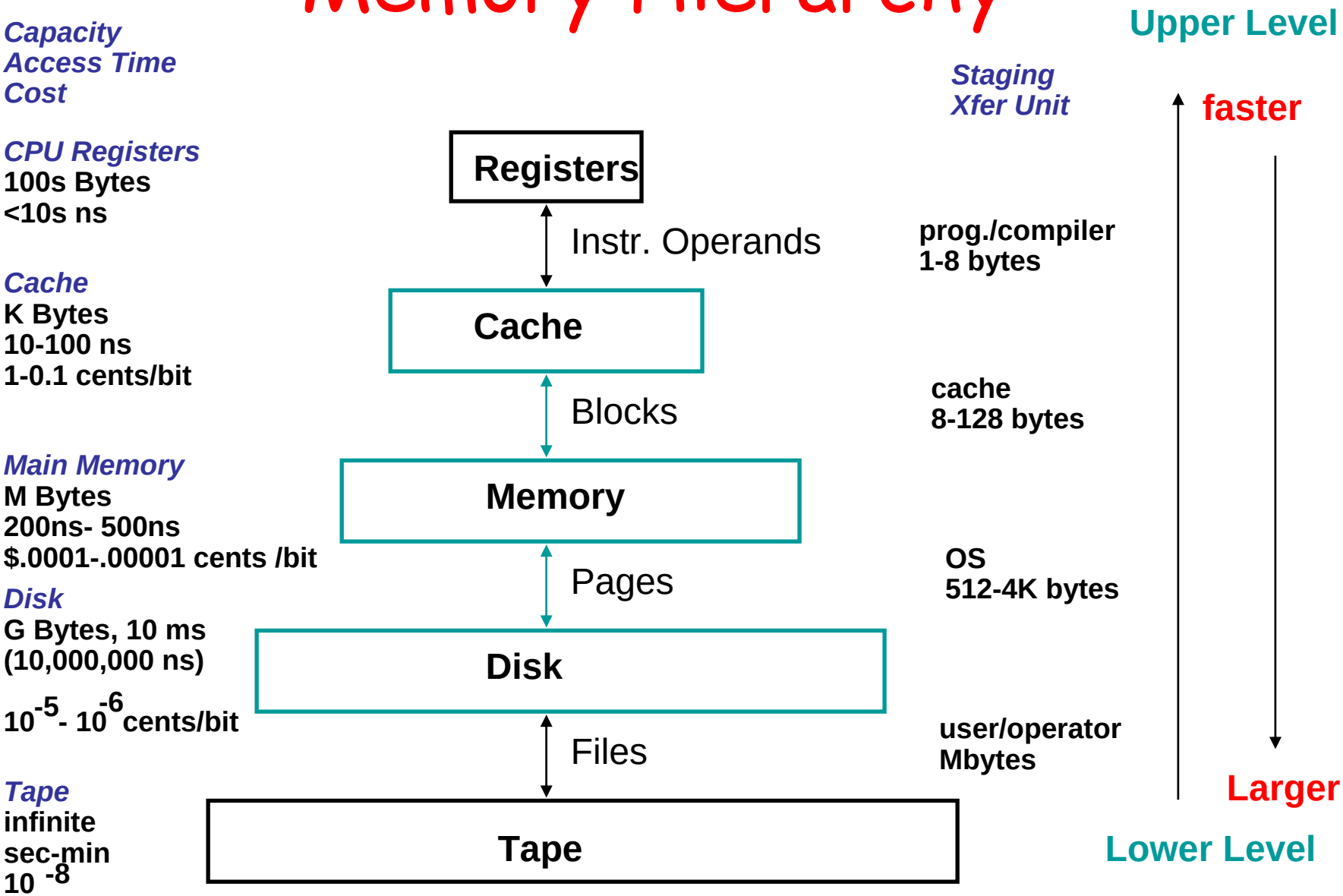
Cache



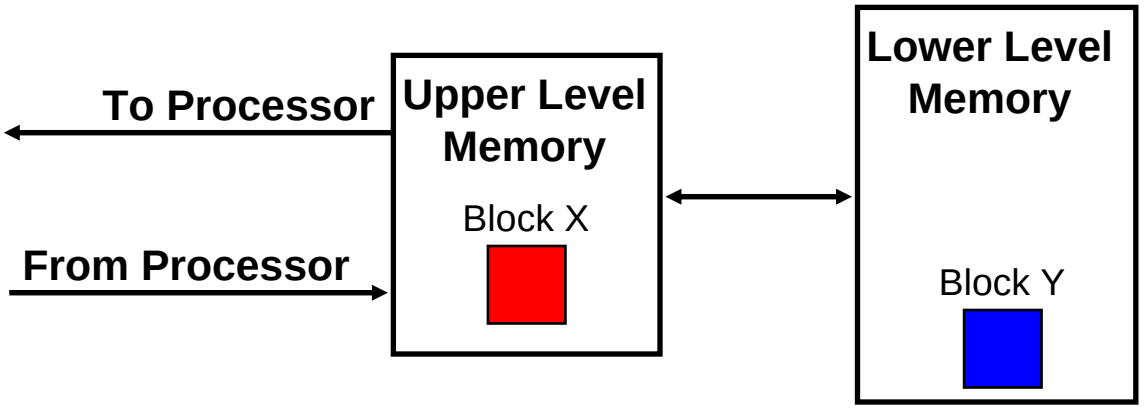
Word transfer

Block transfer

Memory Hierarchy

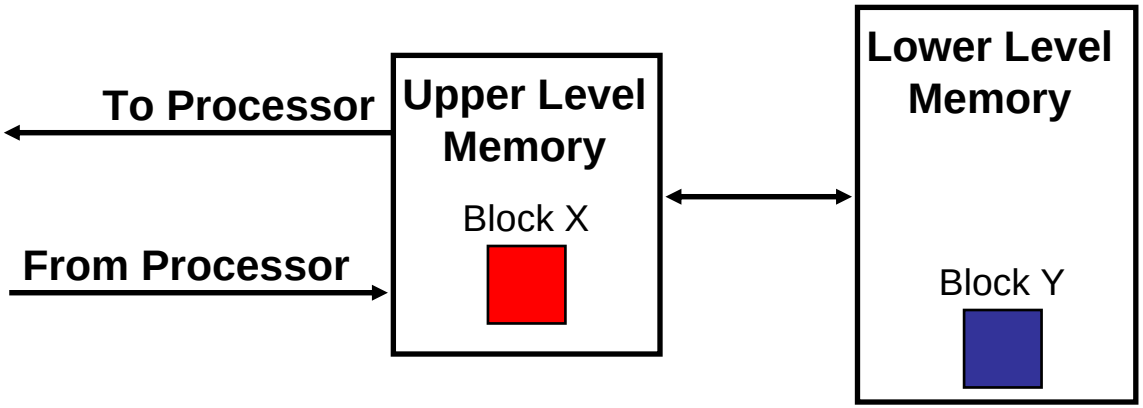


Memory Hierarchy: Principles of Operation



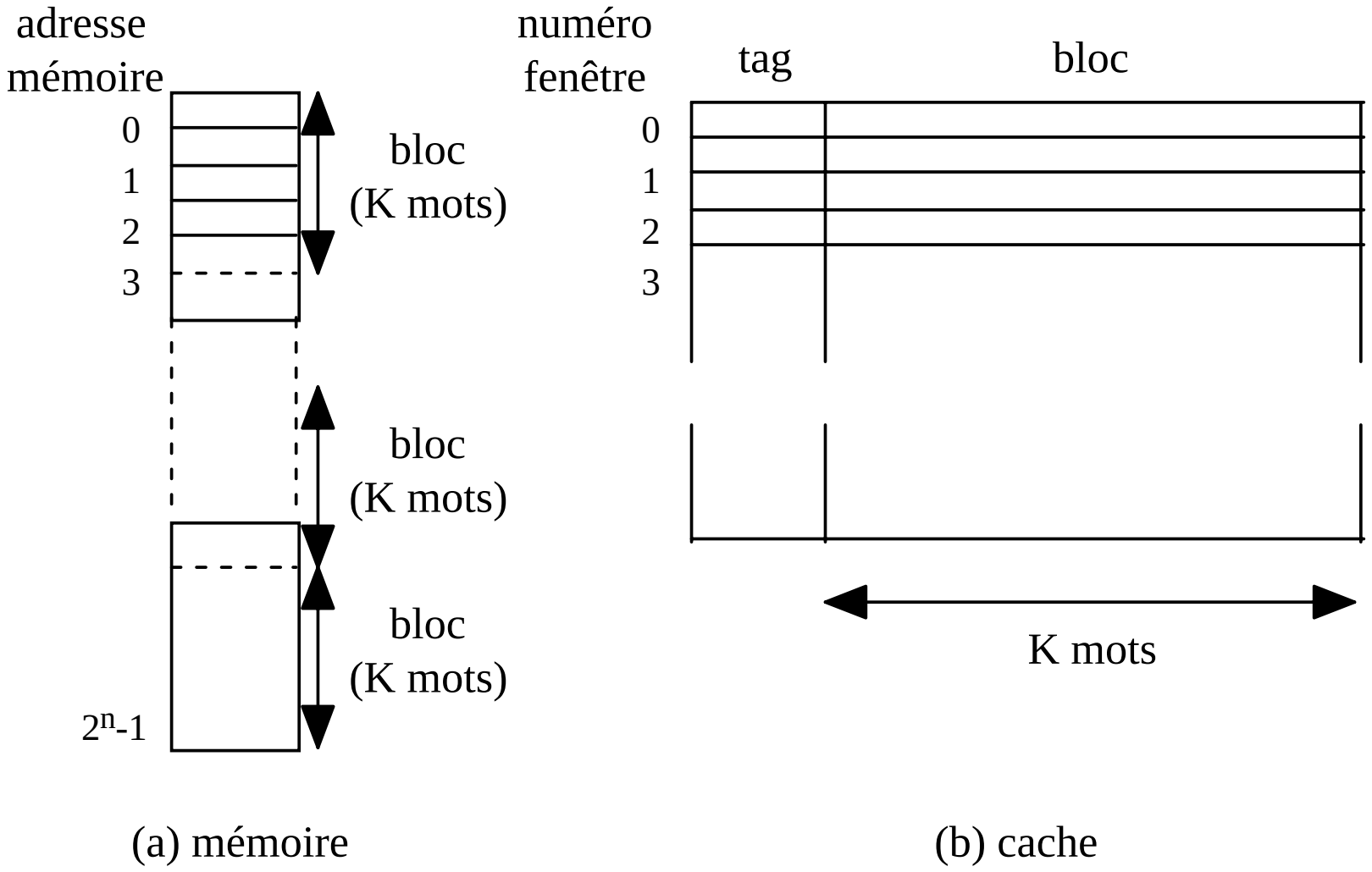
- **At any given time, data is copied between only 2 adjacent levels:**
  - **Upper Level (Cache) : the one closer to the processor**
    - **Smaller, faster, and uses more expensive technology**
  - **Lower Level (Memory): the one further away from the processor**
    - **Bigger, slower, and uses less expensive technology**
- **Block:**
  - **The minimum unit of information that can either be present or not present in the two level hierarchy**

Memory Hierarchy: Terminology



- **Hit**: data appears in some block in the upper level (example: Block X)
  - Hit Rate: the fraction of memory access found in the upper level
  - Hit Time: Time to access the upper level which consists of
    RAM access time + Time to determine hit/miss
- **Miss**: data needs to be retrieve from a block in the lower level (Block Y)
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Time to replace a block in the upper level +
    Time to deliver the block the processor
- Hit Time << Miss Penalty (500 instructions on 21264!)

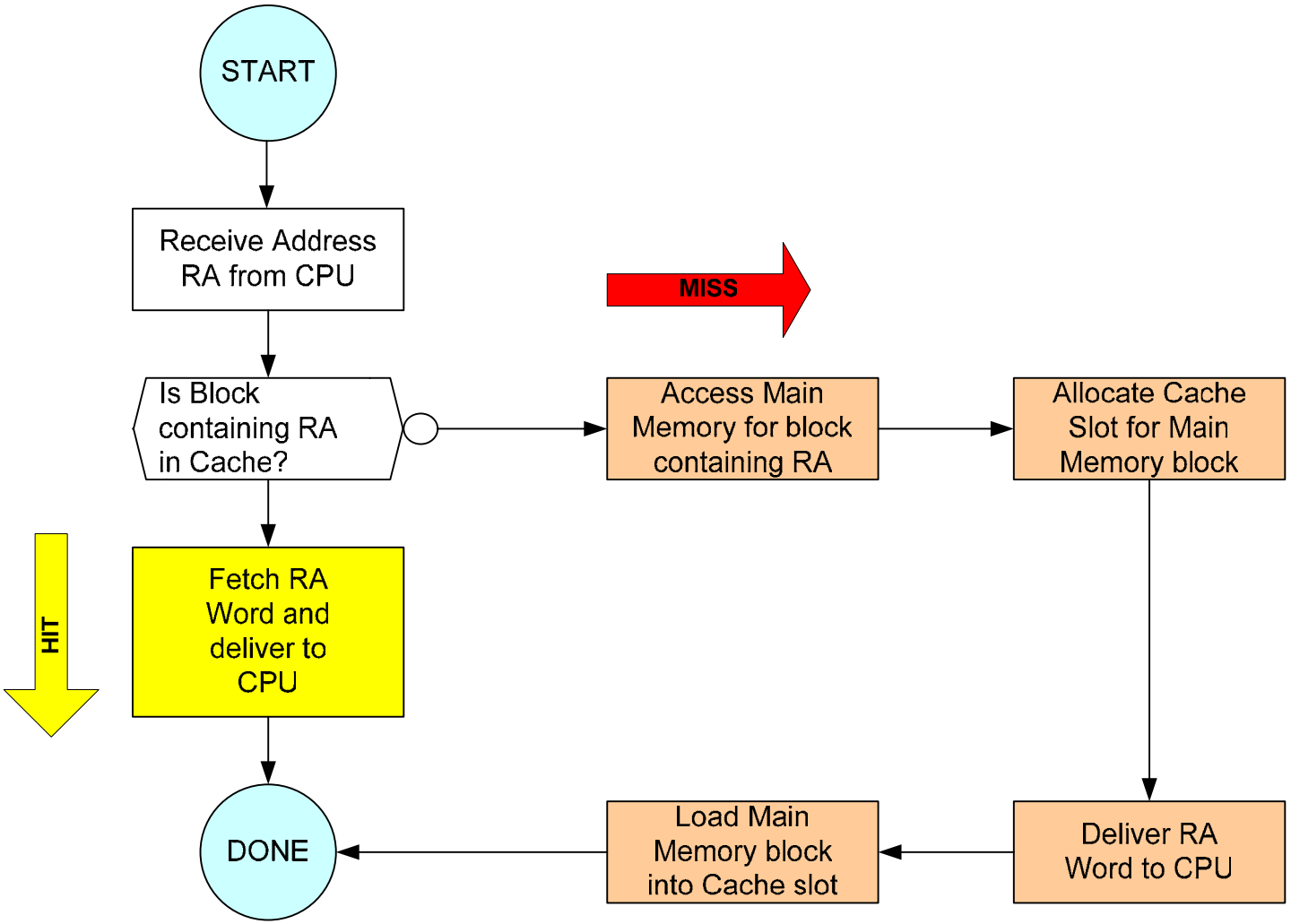
Cache/main memory



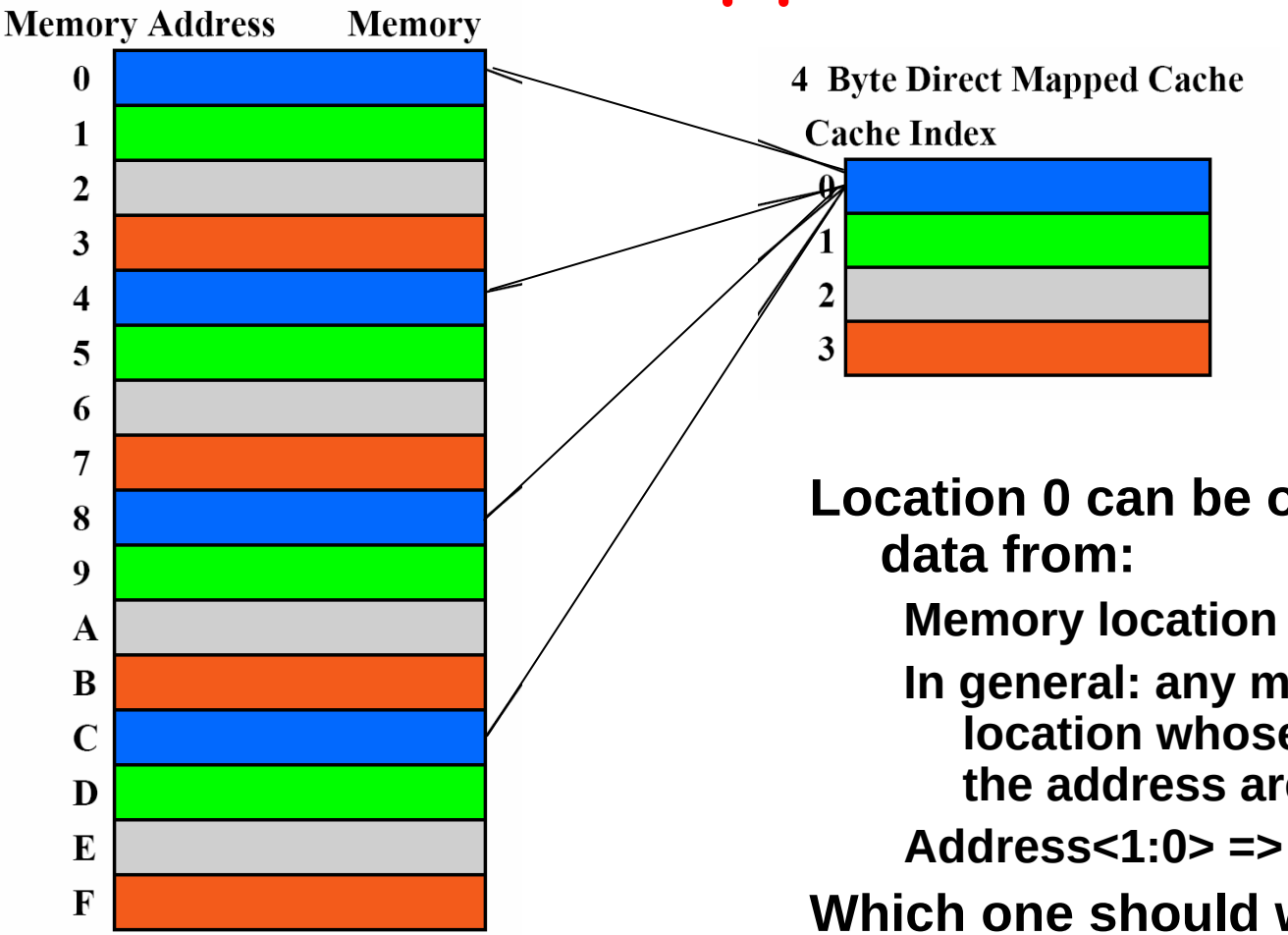
(a) mémoire

(b) cache

Cache Read Operation



Direct Mapped Cached



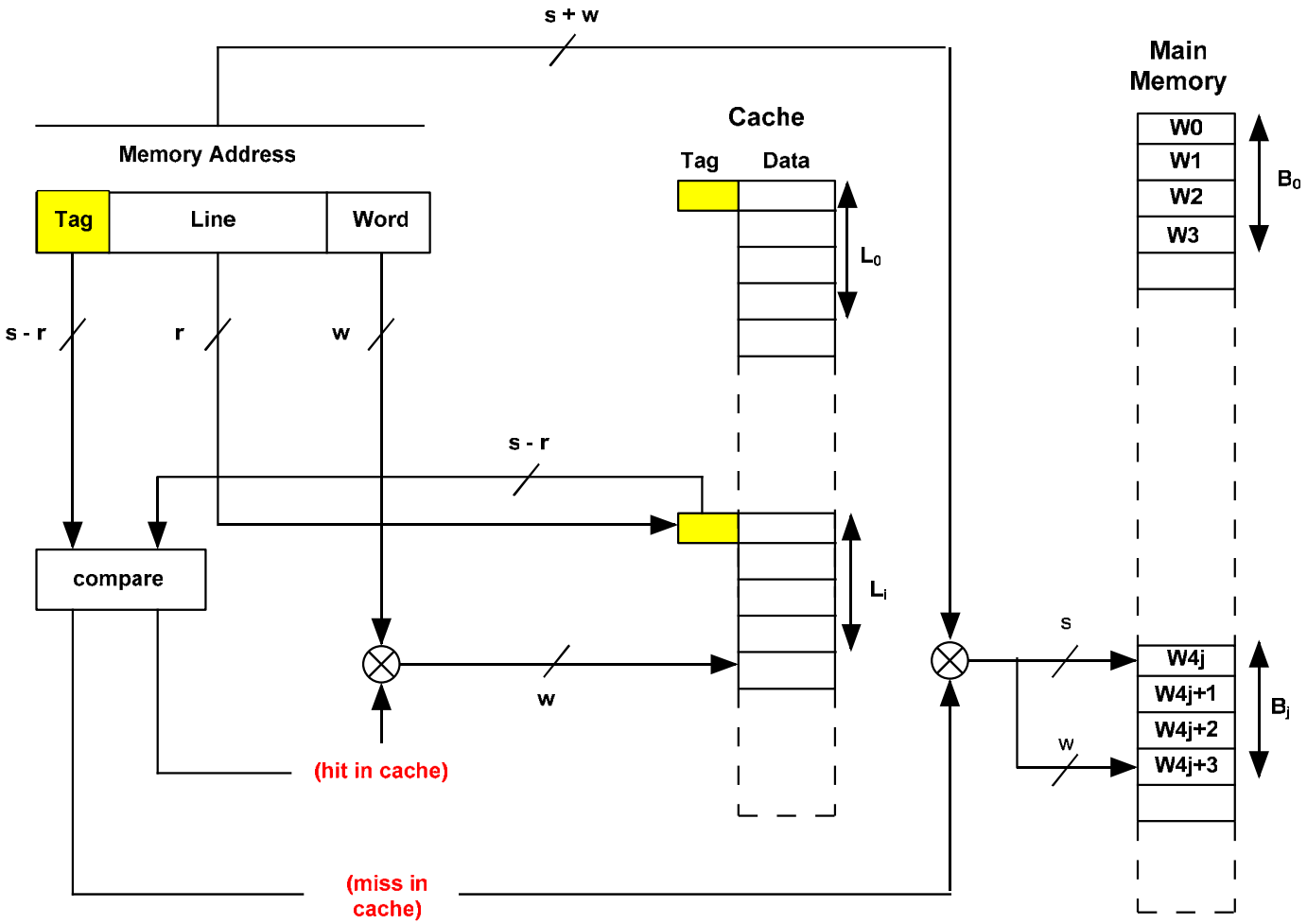
Location 0 can be occupied by data from:

- **Memory location 0, 4, 8, ... etc.**
- **In general: any memory location whose 2 LSBs of the address are 00**
- **Address<1:0> => cache index**

Which one should we place in the cache?

How can we tell which one is in the cache?

Direct-Mapping Cache Organization



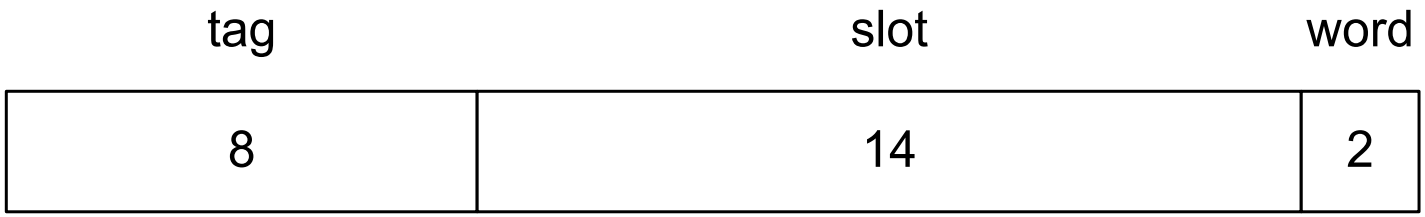
Example of Mapping

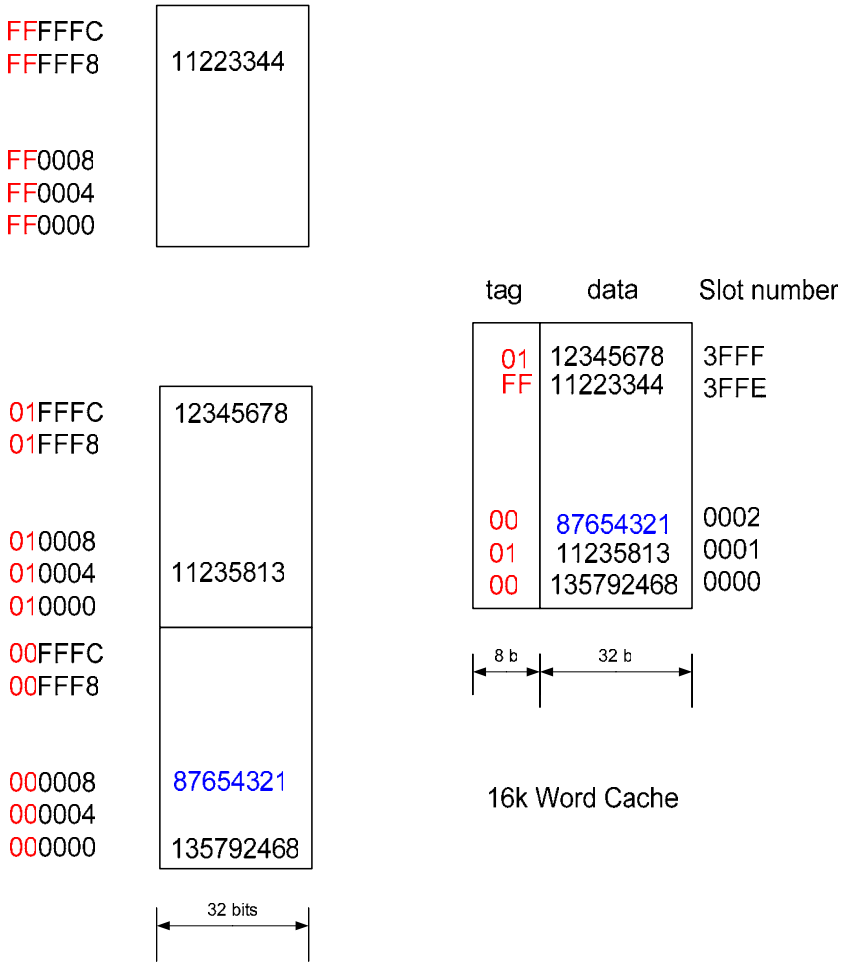
- The Cache can hold 64 KB
- Data is transferred between main memory and the cache in blocks of 4 bytes each.
- The Main Memory consists of 16 MB, with each byte directly addressable by a 24-bit address.
- K = 4 = 2^w; **w** = 2
- m = 64 k / 4 = 16 k = 2^{14} lines in the cache
- **r** = 14
- Main memory: 16 M = 2^{24}; **s** = 22 = 24 - w

Direct Mapped Cached

- j : numéro du bloc en mémoire
- m : nombre de lignes du cache
- Le bloc Bj ne peut être copié que dans la ligne i du cache, avec
- i = j modulo m

Main Memory Address =




FFFFFC
FFFFF8
11223344
FF0008
FF0004
FF0000
01FFFC
01FFF8
12345678
010008
010004
010000
11235813
00FFFC
00FFF8
000008
000004
000000
87654321
135792468
32 bits
16 MB Main Memory
tag
data
Slot number
01
FF
12345678
11223344
3FFF
3FFE
00
01
00
87654321
11235813
135792468
0002
0001
0000
8 b
32 b
16k Word Cache

Access

To select:

Slot ←(address & 0xFFFF)>>2

Tag ←(address & 0xFF0000)>>16

if Cache[Slot].Tag == Tag

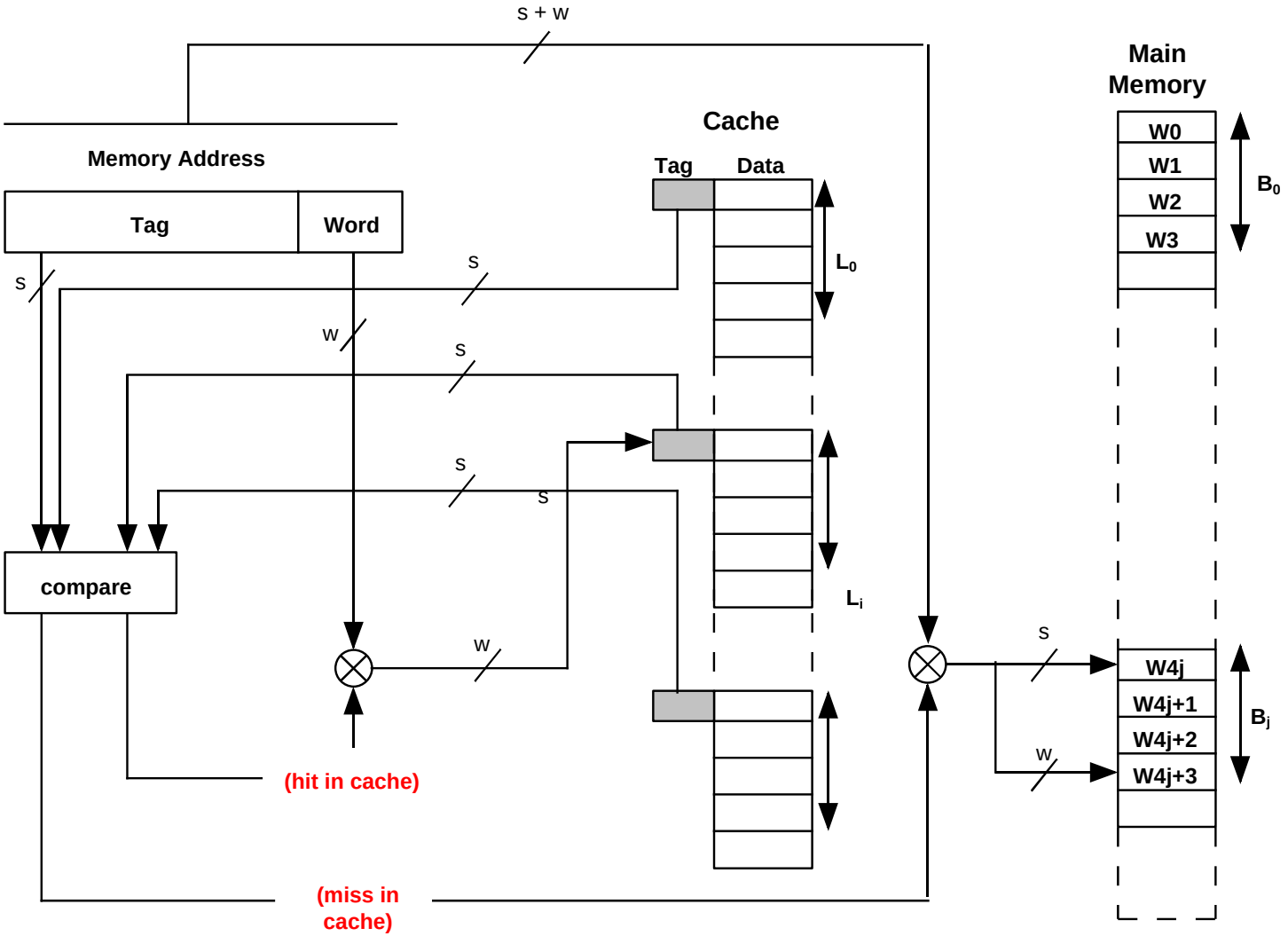
then

    Return Cache[Slot].data

else

    Miss // access to main memory

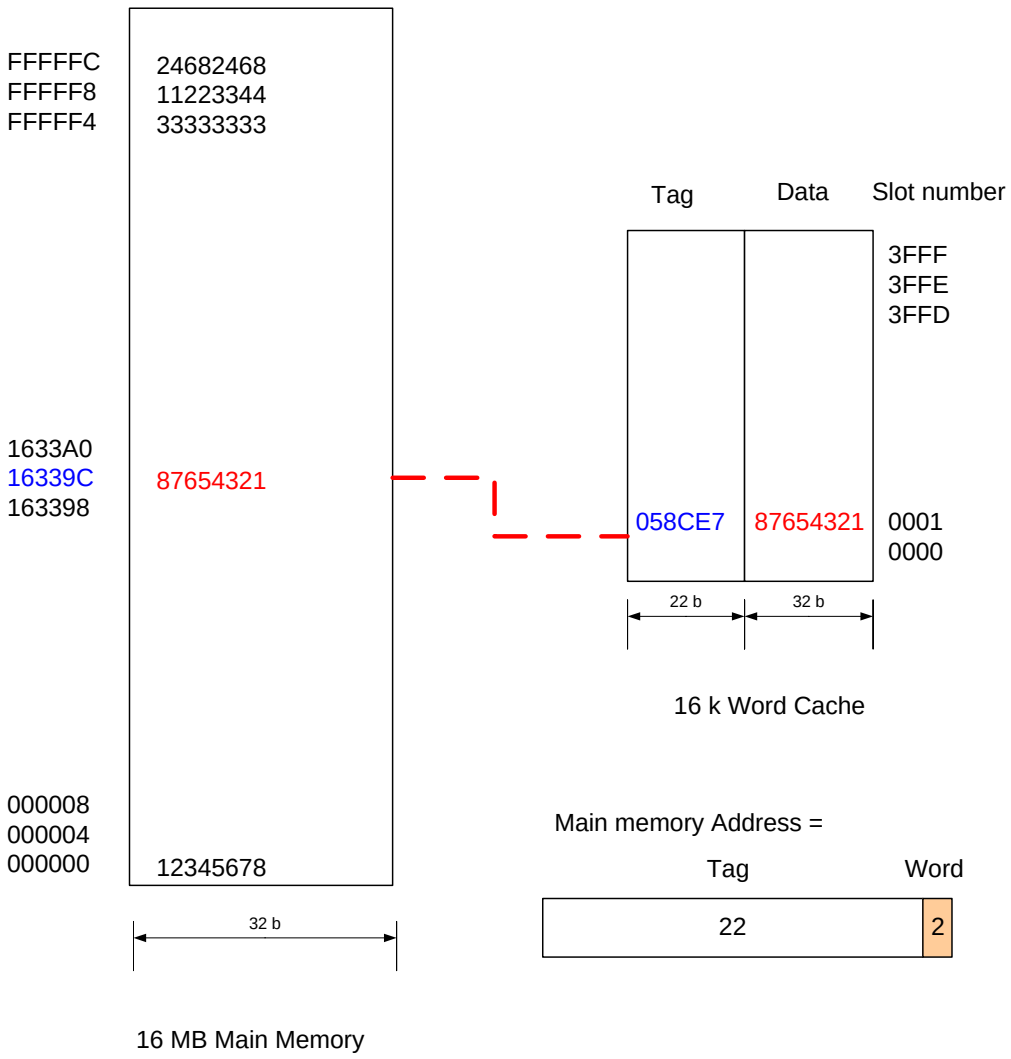
Fully Associative Cache

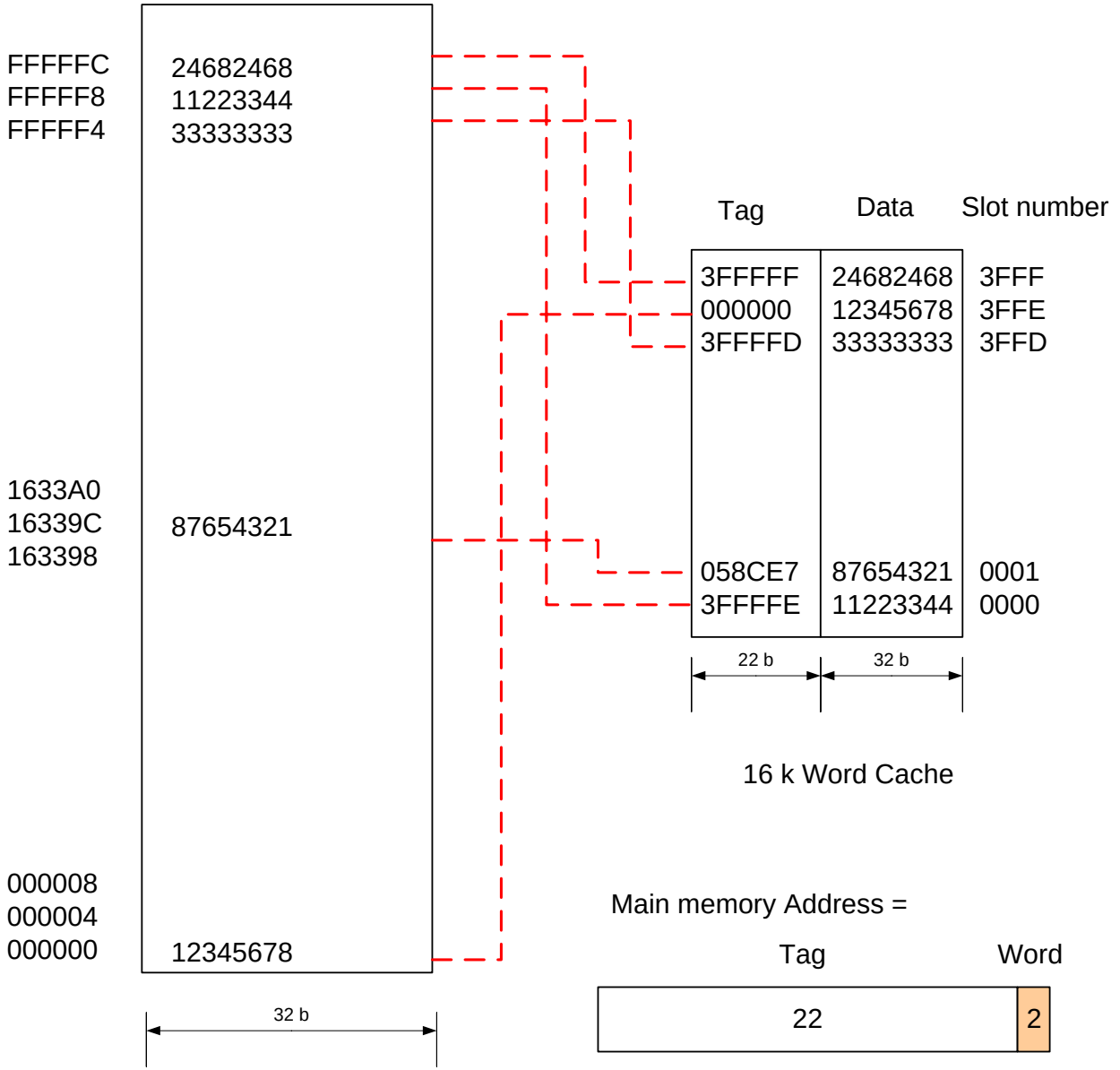


Associative Mapping

- Tag ← address >>2

- Ex: 16339C
- 000101100011 001110011100
- 000001011000 110011100111
- 058CE7



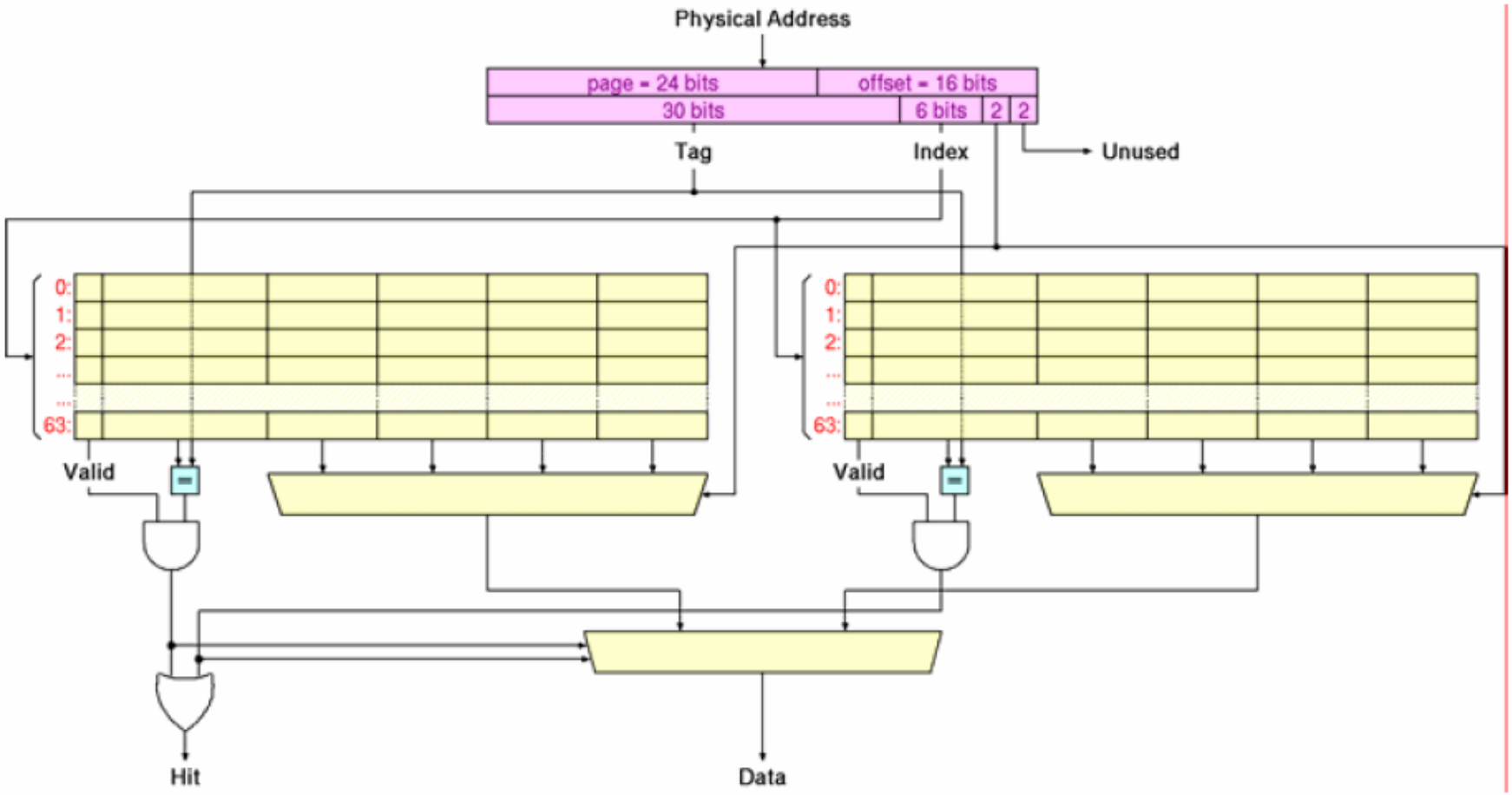

FFFFFC
FFFFF8
FFFFF4
24682468
11223344
33333333
1633A0
16339C
163398
87654321
000008
000004
000000
12345678
32 b
16 MB Main Memory
Tag
Data
Slot number
3FFFFF
000000
3FFFFD
24682468
12345678
33333333
3FFF
3FFE
3FFD
058CE7
3FFFFE
87654321
11223344
0001
0000
22 b
32 b
16 k Word Cache
Main memory Address =
Tag
Word
22
2

Set Associative Cache

- Combination of the previous solutions
- The cache is divided into v sets
- Each of which consists of k lines
- The number of lines in the cache is m=v*k
- Block Bj can be mapped into any of the lines of set i such that
- i = j modulo v

Set Associative Cache: Example

- 2 lines per set



Performances

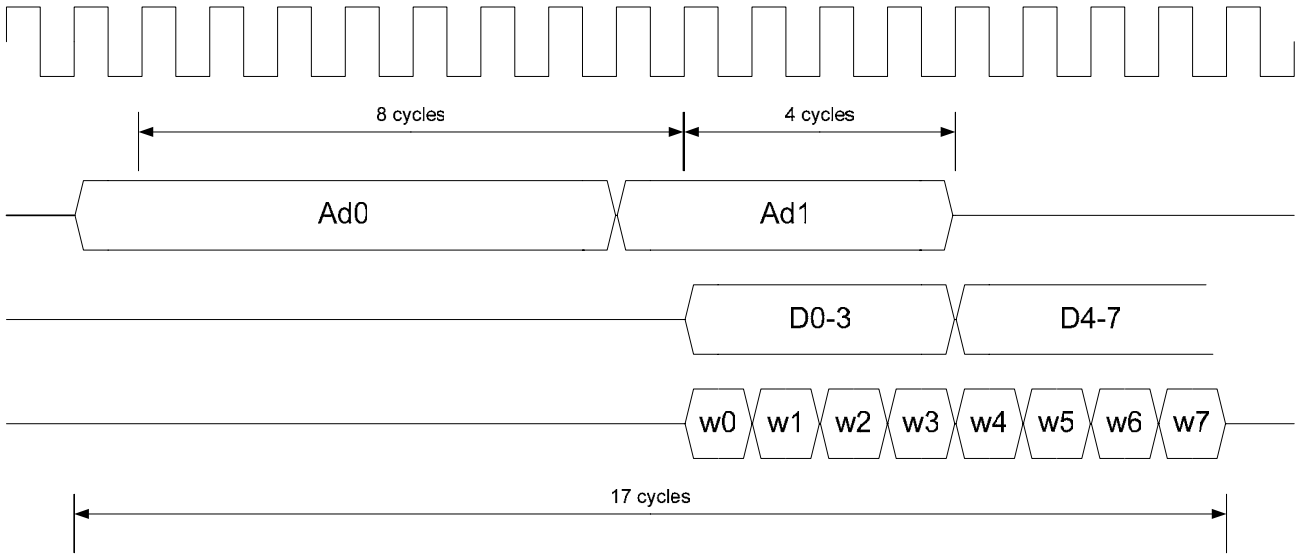
- Ex: cache de blocs de 8 mots
- 1 cycle pour envoyer adresse
- Accès au premier mot: 8 cycles
- Accès aux 7 mots suivants: 4 cycles
- 1 cycle pour transmettre la donnée
- Si une DRAM unique: 1+8+(7x4)+1=38

Performances (2)

4 modules entrelacés

Transfert de 8 mots

1+8+4+4=17 cycles



Performance du cache

- h: hit rate, 1-h: miss rate
- C: time to access information in the cache
- M: miss penalty
- Average access time = hC+(1-h)M
- Ex: h=0.95 pour instructions, O.90 pour données. 30% d'instructions avec accès mémoire.

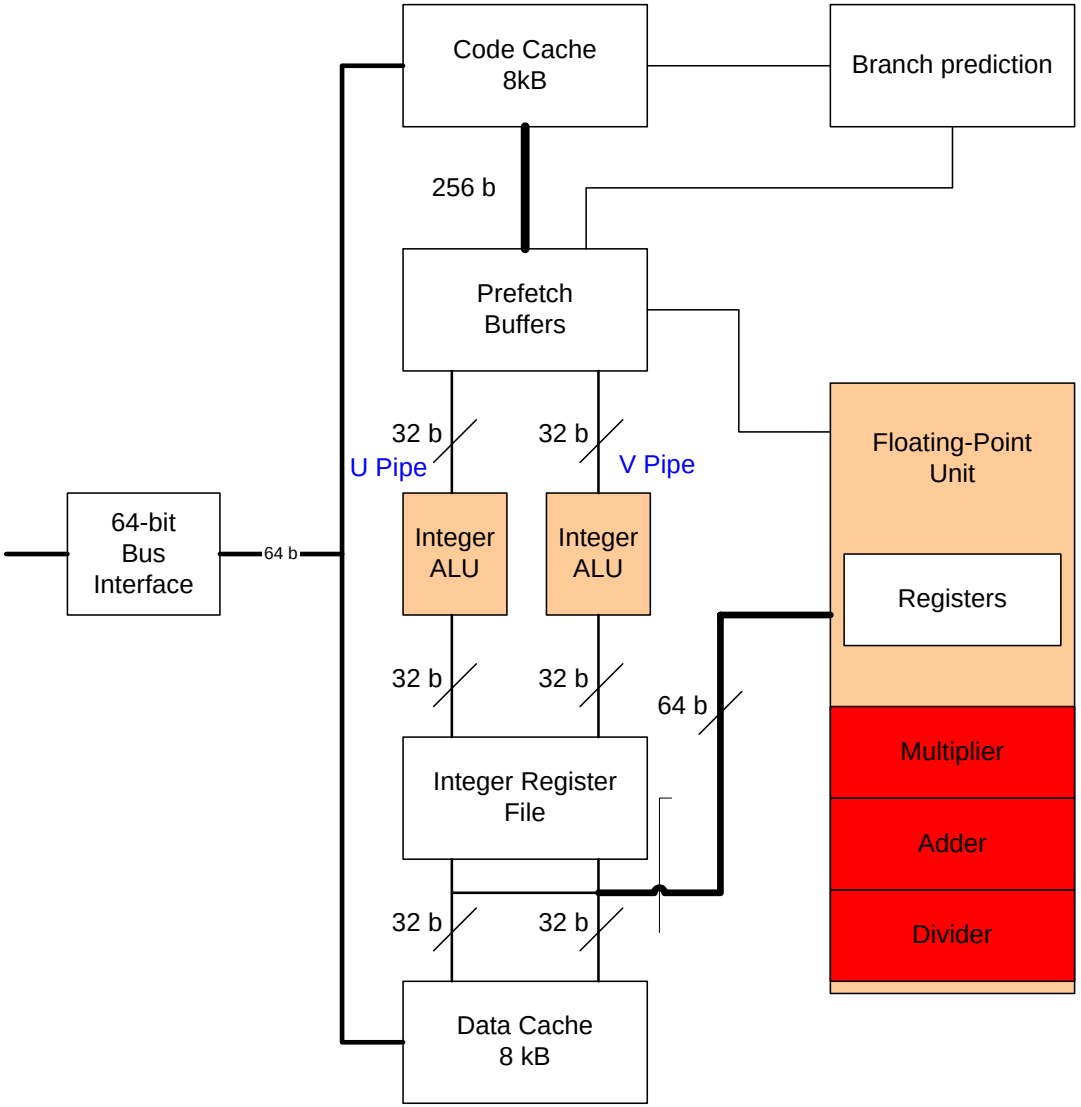
$$\frac{\text{temps sans cache}}{\text{temps avec cache}} = \frac{130 \times 10}{100(0.95 \times 1 + 0.05 \times 17) + 30(0.9 \times 1 + 0.1 \times 17)} = 5.04$$

1+8+1

i-fetch

d-fetch

Pentium


Code Cache
8kB
Branch prediction
256 b
Prefetch
Buffers
32 b
32 b
U Pipe
V Pipe
64-bit
Bus
Interface
64 b
Integer
ALU
Integer
ALU
Floating-Point
Unit
Registers
32 b
32 b
64 b
Multiplier
Integer Register
File
Adder
Divider
32 b
32 b
Data Cache
8 kB

References

- Gershon Kedem, "*Computer Organization and Programming Lecture- 26: Memory, Cache Memory*", Oct. 2004, http://kedem.duke.edu/cps104/Lectures
- [HVZ96] V.C. Hamacher, Z.G. Vranesic, S.G. Zaky. « Computer Organization ». 4th Edition, Mac Graw-Hill, 1996.
- [PH94] D.A. Patterson, J.L. Hennessy. « Computer Organization & Design: the Hardware/Software Interface ». Morgan Kaufmann, 1994.
- [PH96] D.A. Patterson, J.L. Hennessy. « Architecture des Ordinateurs : une approche quantitative ». Thomson Publ., 1996.
- [STA96] W. Stallings. « Computer Organization and Architecture » 4th Edition, Prentice Hall, 1996