

Computer Organization

Improving the Performance



**Department of Computer Science
Missouri University of Science &
Technology
hurson@mst.edu**

Computer Organization

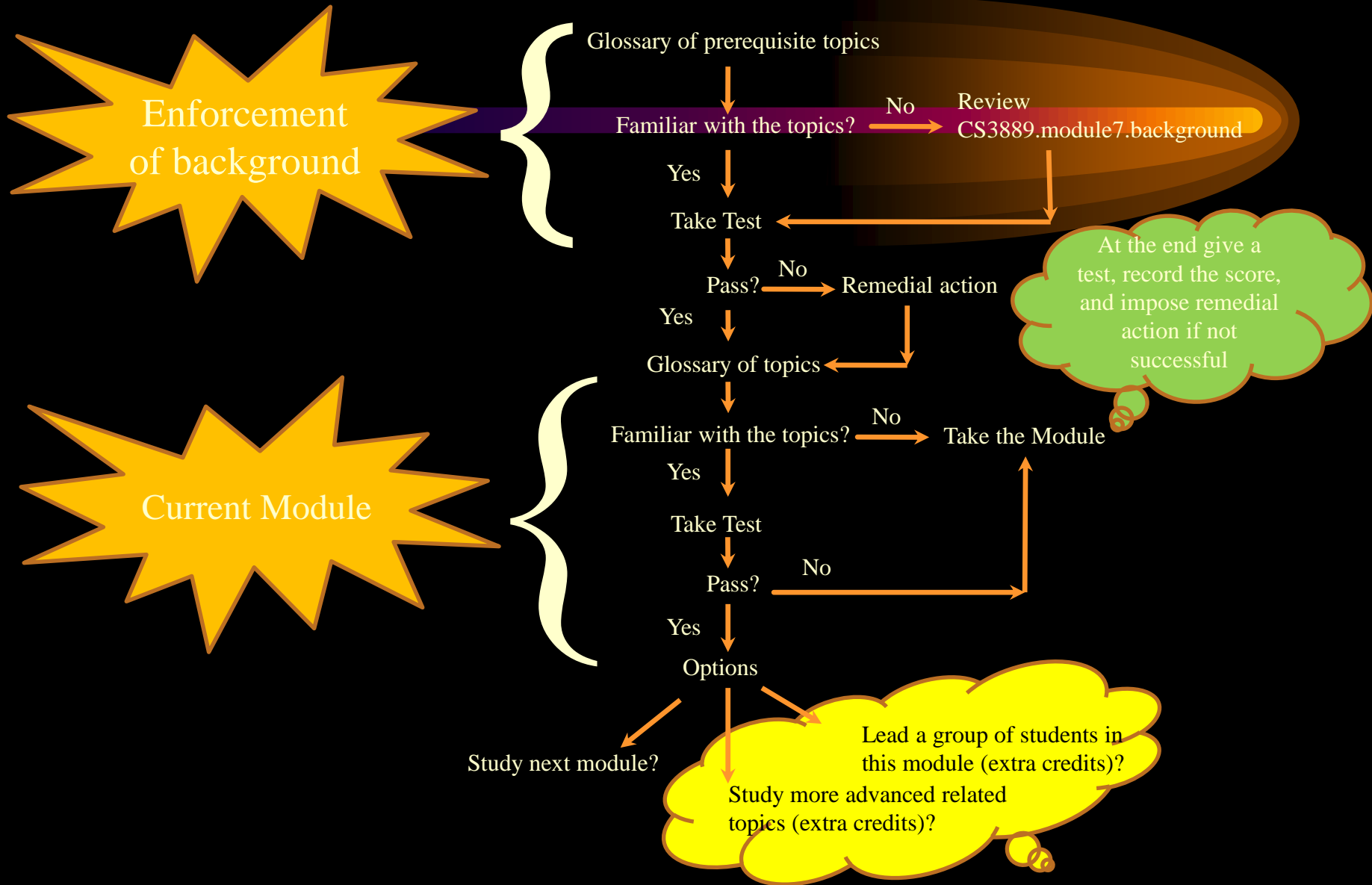


Note, this unit will be covered in five lectures. In case you finish it earlier, then you have the following options:

- 1) Take the early test and start CS3889.module8
- 2) Study the supplement module (supplement CS3889.module7)
- 3) Act as a helper to help other students in studying CS3889.module7

Note, options 2 and 3 have extra credits as noted in course outline.

Computer Organization



Computer Organization



◆ **Computation Gap**

- ★ Computation gap is defined as the difference between computational power demanded by the application environments and computational capability of the existing computers.
- ★ Today, one can find many applications which require orders of magnitude more computations than the capability of the so called **super-computers** and **super-systems**.

Computer Organization

◆ Computation Gap

- ★ Suppose a machine capable of handling 10^6 characters per second is in hand. How long does it take to search 25 terabytes of data?

$$\frac{25 * 10^{12}}{10^6} = 25 * 10^6 \text{ sec} \approx 4 * 10^5 \text{ min.} \approx 7 * 10^3 \text{ Hours} \approx 290 \text{ days}$$

NOT PRACTICAL!

Computer Organization



◆ **Computation Gap – SOLUTIONS?**

- ★ Reduce the amount of needed computations (advances in software technology and algorithms).
- ★ Improve the speed of the computers:
 - **Physical Speed**
 - **Logical Speed**

Computer Organization



◆ **Computation Gap** — Advances in Software Technology and Algorithms

- ★ Since the early days of computers, the development of software support to maximize hardware utility has stimulated much research.
- ★ Software systems were developed to tailor the embedded hardware features of a system to a specific application.
- ★ Various data structure techniques can be used in order to achieve a higher performance.
- ★ Different algorithms can be developed to improve performance.

Computer Organization



◆ **Computation Gap — Advances in Software Technology and Algorithms**

- ★ A compiler equipped with an optimizer routine improves performance during runtime by creating an efficient target language program.
- ★ A vectorized and parallelized compiler can detect the parallelism in an application program and rearrange the instructions in the object program to allow the simultaneous execution of independent instructions or block of instructions on the target machine, during the run time.

Computer Organization

◆ **Computation Gap** — *Advances in Technology*

- ★ Transition from vacuum tubes to VLSI has made it possible to reduce the gate switching delay and size, and to increase the reliability of the hardware components.
- ★ In 1944, a basic operation was executed in 333 msec. About 8 years later, due to the advances in technology, the same basic operation was executed in 282 μ sec. In the early 1960s because of the advances in technology again, it took 300 nsec to perform the same operation.

Computer Organization



◆ **Computation Gap** — *Advances in Technology*

- ★ Is it possible to handle the same basic operation in ≈ 300 pico sec?
- ★ In the past, performance (speed) improvement due to the advances in technology has been at the rate of 10^3 per decade. Should we expect the same improvement rate forever?

Computer Organization

◆ Computation Gap — Advances in Technology

- ★ **Limitations: Speed of Light and Distance:** Light travels $12 * 10^9$ inch per sec. or 12 inch per η sec.
- ★ In 300 pico sec. light travels 4 inches. Therefore, in a hardware unit (basic operation), if the total signal propagation distance is more than 4 inches then it is impossible to execute the same basic operation in 300 pico seconds.

Computer Organization



◆ **Computation Gap** — **Advances in Technology**

- ★ Advances in technology reduce the circuit switching delay and miniaturize the hardware circuits. Nevertheless, it cannot transfer signals faster than the speed of light, and cannot eliminate the distance.
- ★ In the late 1960s Moore predicted that, component density on a chip was quadrupling every three or four years. However, as advances in technology approach the limit, Moore's Law is no longer applicable.

Computer Organization



◆ **Computation Gap – Architectural Advances in the Uniprocessor Organization**

- ★ Organization of the conventional uniprocessor systems can be modified in order to remove the existing bottlenecks.
- ★ **Access Gap** is one of the problems in the von Neumann organization.
- ★ **Access Gap** is defined as the time difference between the **CPU cycle time** and the **main memory cycle time**.

Computer Organization

◆ Memory System

- ★ Based on different parameters memory system can be classified in many ways:
 - By Technology (magnetic memory, semi conductor memory, etc.,)
 - By Access Mode (RAM, DAM, SAM, ROM, CAM, etc.,)
 - By Function (Scratch pad, register, cache, etc.)
- ★ Each category can be sub-classified further. For example, with respect to the access mode **RAM** can be further grouped as 2D, 2 1/2 D or 3D, **ROM** can be grouped as PROM, or EPROM, and **DAM** can be of removable head or fixed head type.

Computer Organization

◆ Memory System — Access Mode

- ★ **Address Accessible Memory:** In this class information is accessed by its address in the memory space:
 - **Random Access Memory (RAM):** Access time is independent of the location of the information.
 - **Sequential Access Memory (SAM):** Access time is a function of the location of the information.
 - **Direct Access Memory (DAM):** Access time is partially independent of and partially dependent on the location of the information.
- ★ **Content Addressable Memory:** In this class information is accessed by its contents (or partial contents).

Computer Organization



◆ **Memory System**

- ★ Many parameters including: **Capacity, cycle time, access time, data rate, word length, cost, ...** have been used for the analysis and evaluation of the memory system.

Computer Organization

◆ Memory System

- ★ **Capacity:** maximum number of bits that can be assembled in one operating memory module.
- ★ **Access time:**
 - **RAM:** Is the time elapse between issuing the address and the availability of the information.
 - **Non-RAM:** Is the time elapse between issuing the address and identifying the information.
- ★ **Data Transfer Rate:** Is the maximum amount of information that can be transferred to or from the main memory in a unit of time.

Computer Organization



◆ **Memory System**

- ★ Access time and capacity are interrelated in an inverse fashion.
- ★ Data transfer rate (bandwidth) is the reciprocal of memory cycle time.

Computer Organization

◆ Memory System

★ How to reduce the main memory bottleneck:

● **Software Solutions:** Devise algorithmic techniques to reduce the number of accesses to the main memory.

● **Hardware Solutions:** Reduce the access gap.

- Advances in technology
- Interleaved memory
- Application of registers
- Cache memory



Computer Organization

◆ Interleaved Memory

- ★ A memory is n -way interleaved if it is composed of n independent modules, and a word at address i is in module number $i \bmod n$.
- ★ This implies consecutive words in consecutive memory modules.
- ★ If the n modules can be operated independently and if the memory bus line is time shared among memory modules then one should expect an increase in bandwidth between the main memory and the CPU.

Computer Organization

◆ Interleaved Memory

- ★ To show the effectiveness of memory interleaving, assume a pure sequential program of n instructions.
- ★ For a conventional system in which main memory is composed of a **single module**, the system has to go through n -fetch cycles and n -execute cycles in order to execute the program.
- ★ For a system in which main memory is composed of m modules, the system executes the same program by executing $\lceil n/m \rceil$ -fetch cycles and n -execute cycles.

Computer Organization



◆ **Interleaved Memory**

- ★ Within the scope of interleaved memory, a **memory conflict** (contention, interference) is defined if two or more addresses are issued to the same memory module.
- ★ In the worst case all the addresses issued are referred to the same memory module.
- ★ In this case the system's performance will be degraded to the level of a single module memory organization.

Computer Organization

◆ Cache Memory — Locality of Reference

- ★ Analysis of a large number of typical programs has shown that most of their execution time is spent in a few main routines that are executed repeatedly. This maybe in the form of a single loop, nested loops, or a few subroutines that repeatedly call each other.
- ★ The main observation is that many instructions in each of a few localized areas of the program are repeatedly executed, while the remainder of the program is accessed relatively infrequently. This phenomenon is referred to as **locality of reference**.

Computer Organization

◆ **Cache Memory — Locality of Reference**

★ Locality comes in tow forms:

- **Temporal Locality:** If an item is referenced, it will tend to be referenced again soon (loops).
- **Spatial Locality:** If an item is referenced, items whose addresses are close by will tend to be referenced soon (sequential nature of most instructions).

Computer Organization

◆ **Cache Memory — Locality of Reference**

- ★ Now, if it can be arranged to have the active segments of a program in a fast memory, then the total execution time can be significantly reduced. Such a fast memory is called a **cache memory**.
- ★ **Cache memory** is a level of memory inserted between the main memory and the CPU.

Computer Organization

◆ Cache Memory

- ★ Due to the economical reason, cache is relatively much smaller than main memory.
- ★ The main memory and the cache are partitioned into **blocks of equal size**. Naturally, because of the size gap between the main memory and the cache, at each moment of time a portion of the main memory is resident in the cache.

Computer Organization

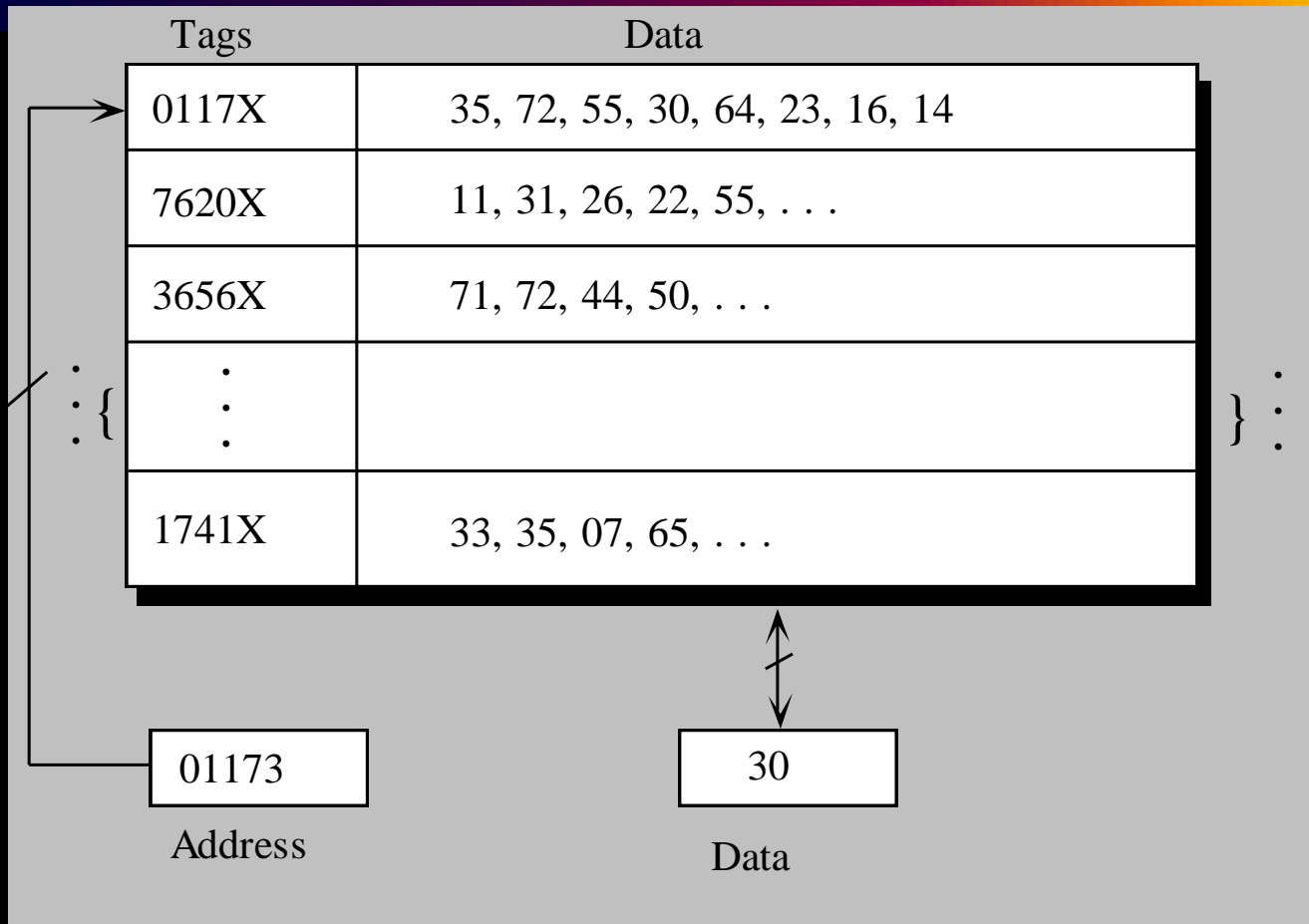


◆ Cache Memory

- ★ Each reference to a memory word is presented to the cache.
- ★ The cache searches its directory:
 - If the item is in the cache, then it will be accessed from the cache.
 - Otherwise, a miss occurs.

Computer Organization

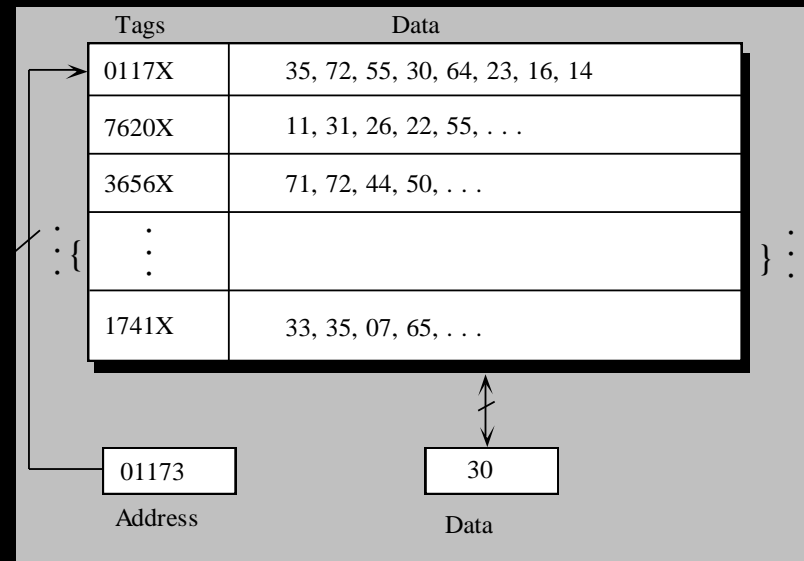
◆ Cache Memory



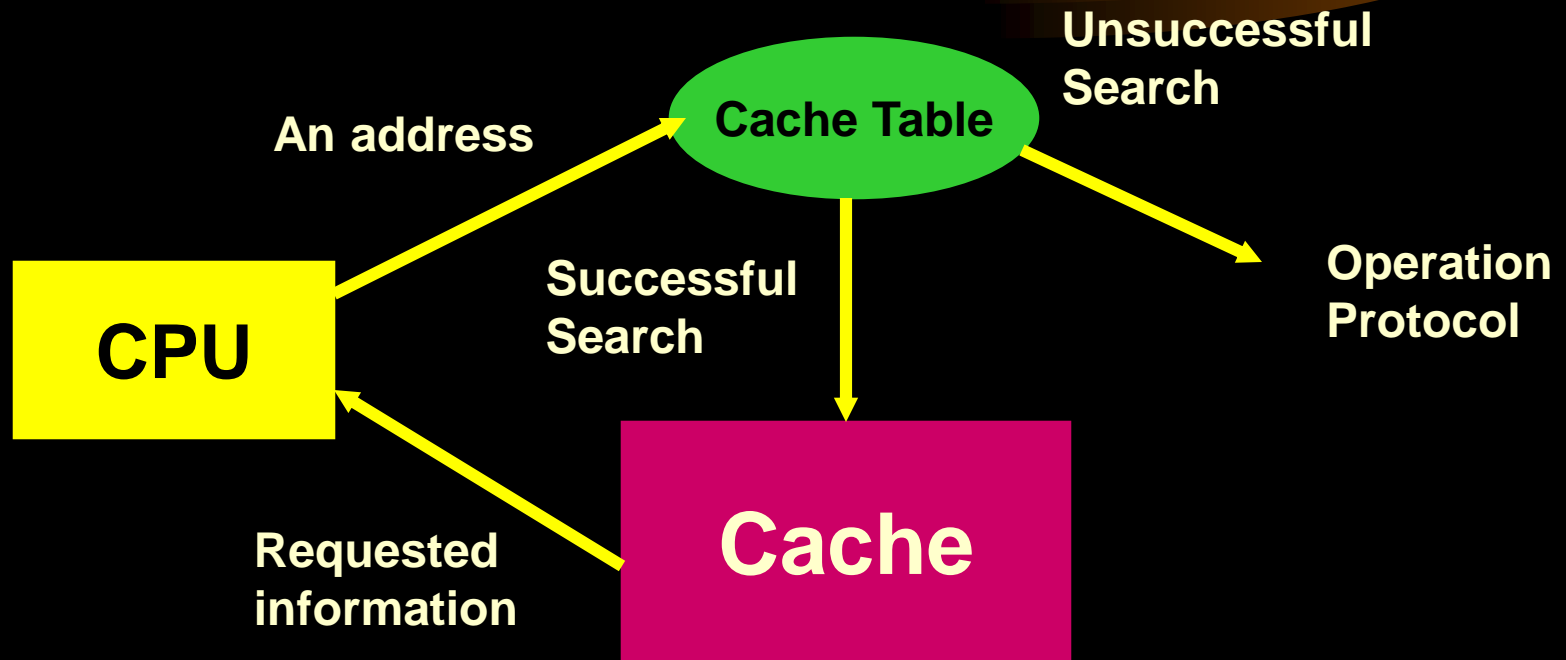
Computer Organization

◆ In our previous diagram:

- ★ A reference to address 01173 is responded by cache.
- ★ A reference to address 01163 produces a miss.



Computer Organization



Computer Organization

◆ Cache Memory — Issues of Concern

* Read Policy

- Load Through

* Write policy (on hit)

- Write through
- Write back \Rightarrow dirty bit

* Write policy (on miss)

- Write allocate
- No-write allocate

* Placement/replacement policy

* Address Mapping

Computer Organization

◆ Cache Memory — Replacement Policy

- ★ For each read operation that causes a cache miss, the item is retrieved from the main memory and copied into the cache. This forces some other item in cache to be identified and removed from the cache to make room for the new item (if cache is full).
- ★ The collection of rules which allows such activities is referred to as the **Replacement Algorithm**.

Computer Organization



◆ Cache Memory — Replacement Policy

- ★ The cache-replacement decision is critical, a good replacement algorithm, naturally, can yield somewhat higher performance than can a bad replacement algorithm.

Computer Organization



◆ Cache Memory — Address Mapping

- Direct Mapping
- Associative Mapping
- Set Associative Mapping

Computer Organization

◆ Cache Memory — Address Mapping

★ The following discussion assumes:

- B = block size (2^b)
- C = number of blocks in cache (2^c)
- M = number of blocks in main memory (2^m)
- S = number of sets in cache (2^s)

Computer Organization

◆ Cache Memory – Direct Mapping

- ★ Block K of main memory maps into block $(K \text{ modulo } C)$ of the cache.
- ★ Since more than one main memory block is mapped into a given cache position, contention may arise even when the cache is not full.

Computer Organization



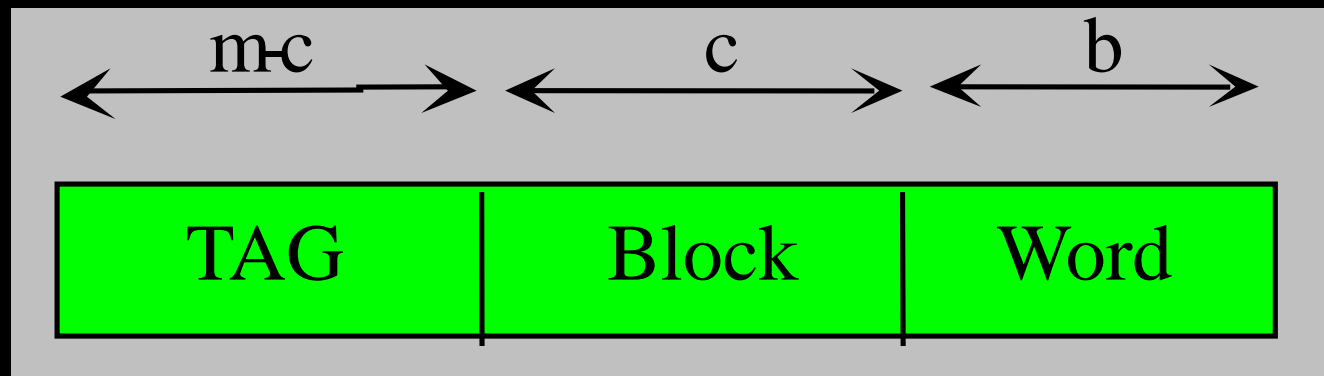
◆ Cache Memory – Direct Mapping

- ★ Address mapping can be implemented very easily.
- ★ Replacement policy is very simple and trivial.
- ★ In general, cache utilization is low.

Computer Organization

◆ Cache Memory – Direct Mapping

- ★ Main memory address is of the following form:



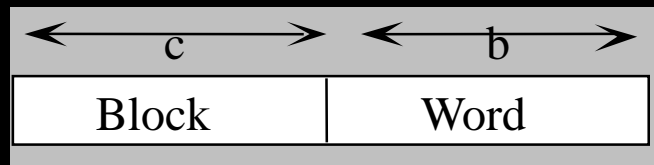
- ★ A **Tag-register** of length $m-c$ is dedicated to each cache block.

Computer Organization

◆ Cache Memory – Direct Mapping

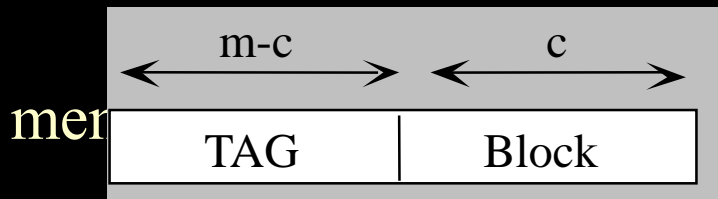
★ Content of $(\text{tag-register})_c$ is compared against the tag portion of the address:

★ If match then hit; and access information at address



from the cache.

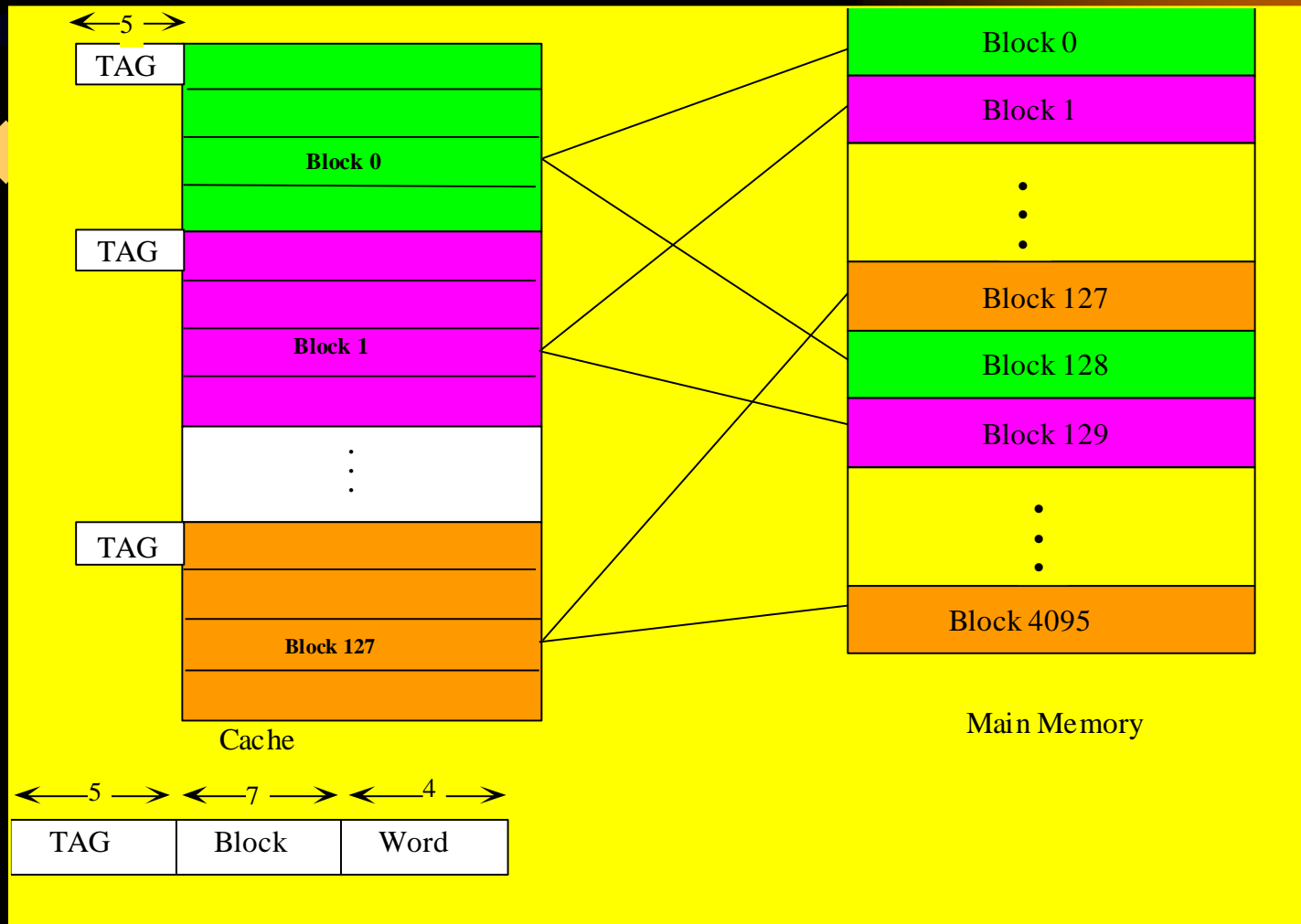
★ If no-match, then miss-hit; bring block



from main

block c of cache.

Computer Organization



Computer Organization

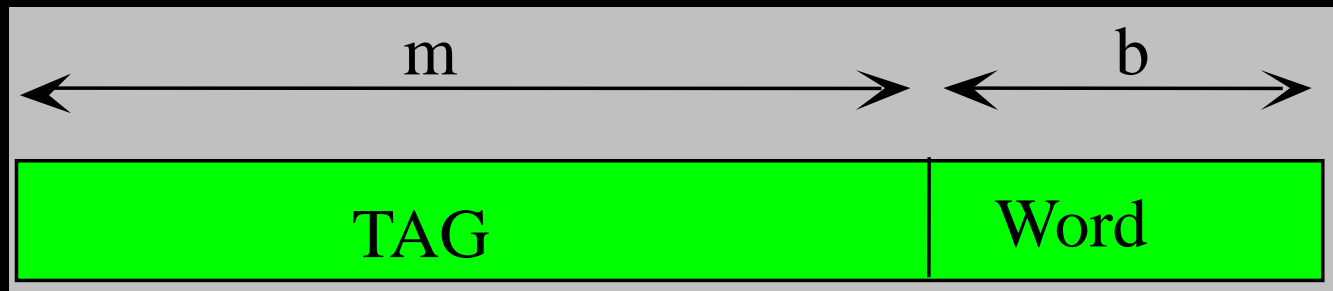
◆ Cache Memory – Associative Mapping

- ★ A block of main memory can potentially reside in any cache block position. This flexibility can be achieved by utilizing a wider Tag-Register.
- ★ Address mapping requires hardware facility to allow simultaneous search of tag-registers.
- ★ A reasonable replacement policy can be adopted (least recently used).
- ★ Cache can be used very effectively.

Computer Organization

◆ Cache Memory — Associative Mapping

- ★ Main memory address is of the following form:

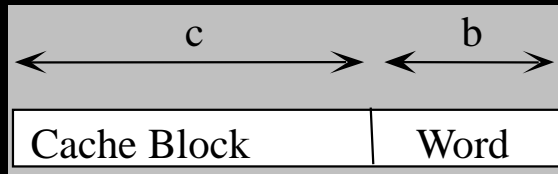


- ★ A tag-register of length m is dedicated to each cache block.

Computer Organization

◆ Cache Memory — Associative Mapping

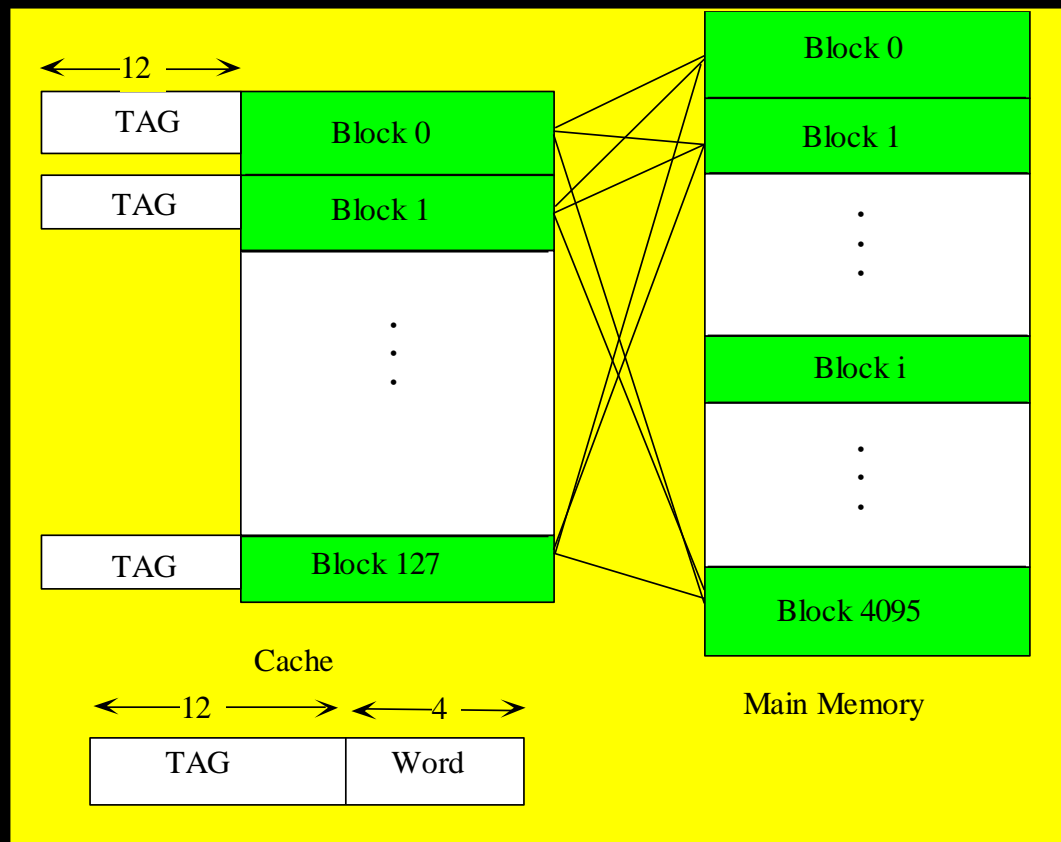
- ★ Contents of Tag portion of the address is searched (in parallel) against the contents of the Tag-registers:
- ★ If match, then hit; access information at address from the cache.



- ★ If no-match, then miss-hit; bring block from memory into the **proper** cache block.

Computer Organization

◆ Cache Memory — Associative Mapping



Computer Organization

◆ Cache Memory — Set Associative Mapping

- ★ Is a compromise between Direct-Mapping and Associative Mapping.
- ★ Blocks of cache are grouped into **sets** (S), and the mapping allows a block of main memory (K) to reside in any block of the set (K modulo S).
- ★ Address mapping can be implemented easily at a more **reasonable hardware cost** relative to the associative mapping.

Computer Organization



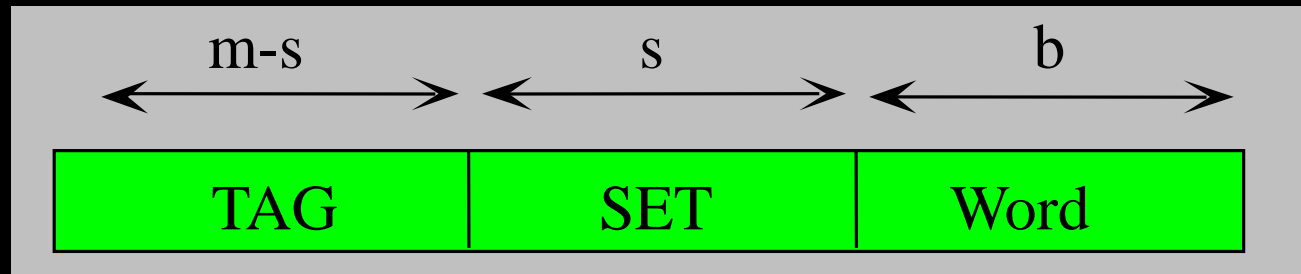
◆ Cache Memory — Set Associative Mapping

- ★ This scheme allows one to employ a reasonable replacement policy within the blocks of a set and hence offers **better cache utilization** than the direct-mapping scheme.

Computer Organization

◆ Cache Memory — Set Associative Mapping

- ★ Main memory address is of the following form:



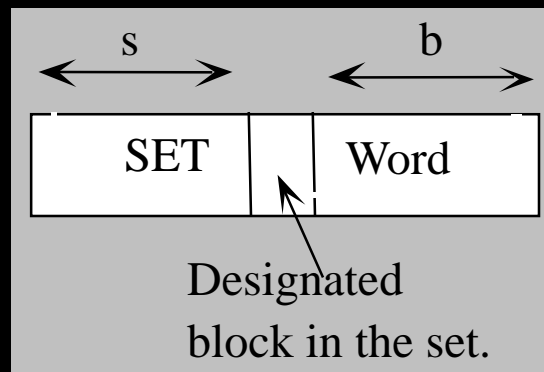
- ★ A tag-register of length $m-s$ is dedicated to each block in the cache.

Computer Organization

◆ Cache Memory — Set Associative Mapping

★ Contents of Tag-registers_s are compared simultaneously against the tag portion of the address:

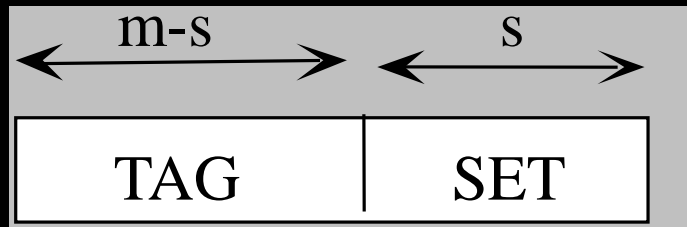
★ If match, then hit; access information at address from the cache.



Computer Organization

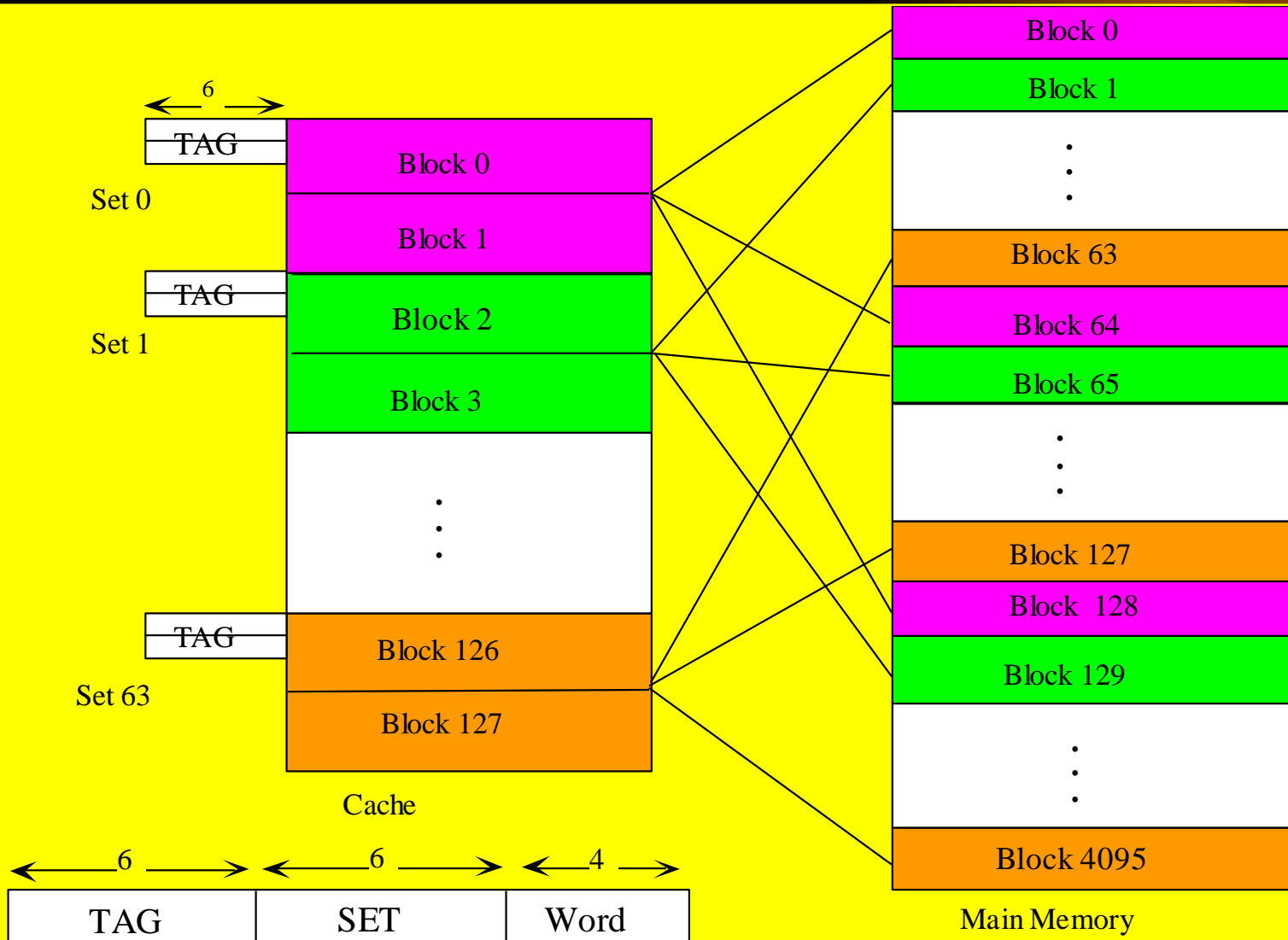
◆ Cache Memory — Set Associative Mapping

★ If no-match, then miss-hit; bring block



from the main memory into the proper block of set s of the cache.

Computer Organization



Computer Organization

◆ Cache Memory — Hit ratio

- ★ Let h be the probability of a cache hit — **hit ratio**

$$h = \frac{\text{\# of accesses responded by cache}}{\text{Total \# of accesses to the memory}}$$

and t_{cache} and t_{main} be the respective cycle times of cache and main memory then:

$$t_{\text{eff}} = t_{\text{cache}} + (1-h)t_{\text{main}}$$

- ★ $(1-h)$ is the probability of a miss — **miss ratio**.

Computer Organization



◆ **Computation Gap** — System Architecture

- ★ To overcome the technological limitations, computer designers have long been attracted to techniques that are classified under the term "concurrency".

Computer Organization



◆ Computation Gap — System Architecture

★ **Concurrency** is a generic term that defines the ability of computer hardware to simultaneously execute many actions at any instant. Within this general term are several well recognized techniques such as Parallelism, Pipelining and Multiprocessing.

Computer Organization

◆ Computation Gap — System Architecture

- ★ Although these techniques have the same origin and are often hard to distinguish, in practice they are different in their general approach.
- ★ In **parallelism** concurrency is achieved by replicating the hardware structure many times, while **pipelining** takes the approach of splitting the function to be performed into smaller pieces and allocating separate hardware to each piece.

Computer Organization

◆ Pipeline Systems

- ★ The term **pipelining** refers to a design technique that introduces concurrency by taking a basic function to be involved repeatedly in a process and partitioning it into several subfunctions with the following properties:
 - Evaluation of the basic function is equivalent to some sequential evaluation of the subfunctions.
 - Other than the exchange of inputs and outputs, there is no interrelationships between subfunctions.
 - Hardware may be developed to execute each subfunction.
 - The execution times of these hardware units are usually approximately equal.

Computer Organization



◆ Pipeline Systems

- ★ Under the aforementioned conditions, the speed up from pipelining equals the number of pipe stages.
- ★ However, stages are rarely balanced and furthermore, pipelining does involve some overhead.

Computer Organization



◆ Pipeline Systems

★ The concept of pipelining can be implemented at different levels. With regard to this issue, one can then address:

- Arithmetic Pipelining
- Instruction Pipelining
- Processor Pipelining

Computer Organization

◆ Pipeline Systems

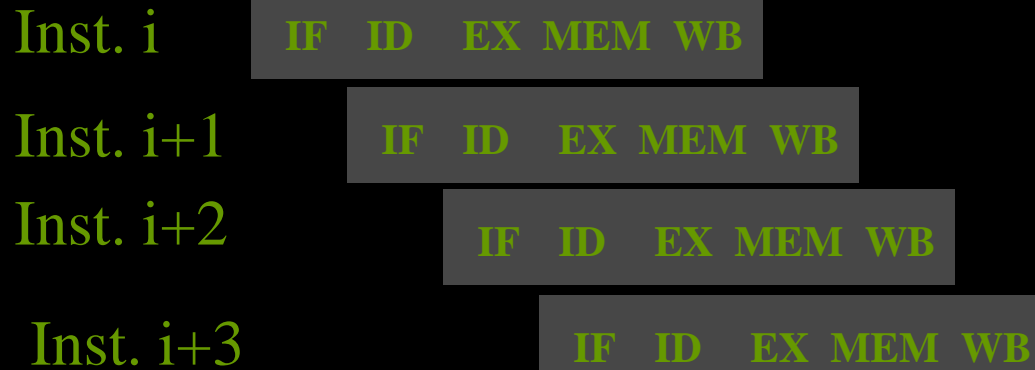
★ Non-pipelined instruction cycle:



Computer Organization

◆ Pipeline Systems

★ Pipelined instruction cycle:



A pipelined instruction cycle gives a peak performance of one instruction every step.

Computer Organization

◆ Pipeline Systems — Example

- ★ Assume a non-pipeline machine has a 10 ns clock cycles. It requires four clock cycles for the ALU and branch operations and five clock cycles for the memory reference operations. Calculate the **average instruction execution time**, if the relative frequencies of these operations are 40%, 20%, and 40%, respectively.

$$\text{Ave. instr. exec. time} = 10 * [(40\%+20\%) * 4 + 40\% * 5] = 44 \text{ ns}$$

Computer Organization

◆ Pipeline Systems — Example

- ★ Now assume we have a pipeline version of this machine. Furthermore, due to the clock skew and set up, pipelining adds 1 ns overhead to the clock time. Ignoring the latency, now calculate the **average instruction execution time**.

Ave. instr. exec. time = $10 + 1$ ns, and

Speed up = $44/11 = 4$

Computer Organization

◆ Pipeline Systems — Example

- ★ Assume that the time required for the five units in an instruction cycle are, 10 ns, 8 ns, 10 ns, 10 ns, and 7 ns. Further, assume that pipelining adds 1 ns overhead. Find the speed up factor:

$$\text{Ave. instr. exec. time}_{\text{unpipeline}} = 10 + 8 + 10 + 10 + 7 = 45 \text{ ns}$$

$$\text{Ave. instr. exec. time}_{\text{pipeline}} = 11 \text{ ns}$$

$$\text{Speed up} = 45/11 = 4.1$$

Computer Organization



◆ Pipeline Systems

- ★ A concept known as **hazard** is a major concern in a pipeline organization.
- ★ A **hazard** prevents the pipeline from accepting data at the maximum rate that the staging clock might support.

Computer Organization



◆ Pipeline Systems

★ A **hazard** can be of three types:

- **Structural Hazard:** Arises from resource conflicts when the hardware cannot support all possible combinations of instructions in simultaneous overlapped execution — two different pieces of data attempt to use the same stage at the same time.

Computer Organization



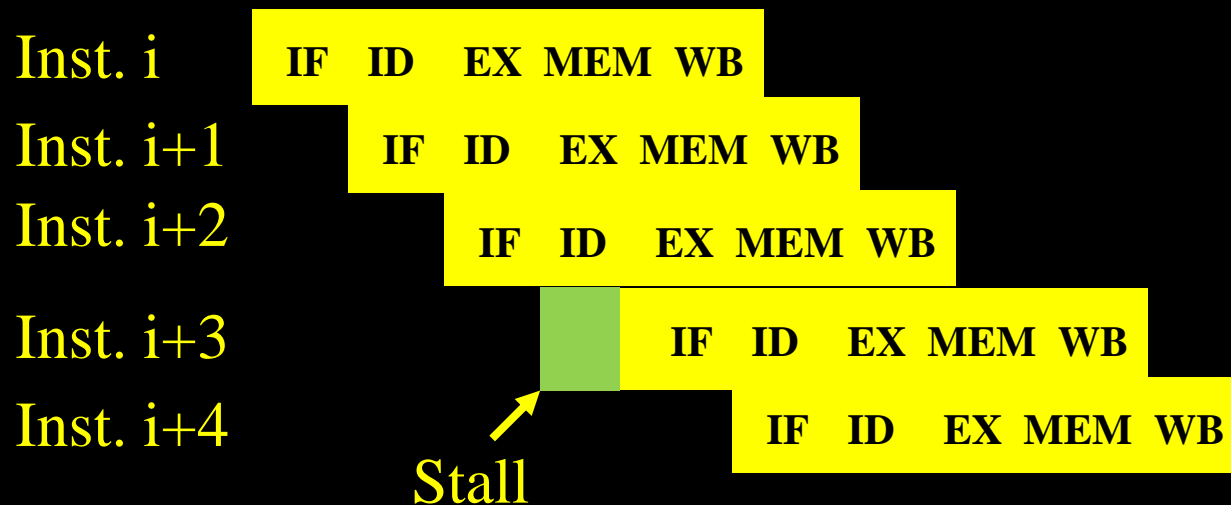
◆ Pipeline Systems

- ★ **Data-Dependent Hazard:** Arises when an instruction depends on the result of a previous instruction — the pass through a stage is a function of the data value.
- ★ **Control Hazard:** Arises from the pipelining of instructions that affect PC — Branch.

Computer Organization

◆ Pipeline Systems

★ **Structural Hazard** (Assume a Single memory pipeline system):



Computer Organization

◆ Pipeline Systems

★ Data Hazard

- A data hazard is created whenever there is a dependence between instructions, and they are close enough that the overlap caused by pipelining would change the order of access to an operand.

```
ADD  R1, R2, R3
SUB  R4, R1, R5
```

Computer Organization

◆ Pipeline Systems

★ Data Hazard — Classification

- Assume i and j are two instructions and j is the successor of i , then one could expect three types of data hazard:
 - Read after write (RAW)
 - Write after write (WAW)
 - Write after read (WAR)

Computer Organization

◆ Pipeline Systems

★ Data Hazard — Classification

- Read after write (RAW) — j reads a source before i writes it (flow dependence).
- Write after write (WAW) — j writes into the same destination as i does (output dependence).

LW $R_1, 0(R_2)$

Add R_1, R_2, R_3

IF ID EX MEM₁ MEM₂ WB

IF ID EX WB

Computer Organization

◆ Pipeline Systems

★ Data Hazard — Classification

- Write after read (WAR) — j writes into a source of i (anti dependence).

SW $0(R_1), R_2$

IF ID EX MEM₁ MEM₂ WB

Add R_2, R_4, R_3

IF ID EX WB

Computer Organization

◆ Pipeline Systems

★ Data Hazard — Example

- Assume 30% of the instructions are **load** and half the time the instruction following a load instruction depends on the result of the load. If the hazard creates a **single-cycle** delay, how much faster is the ideal pipelined machine?

$$CPI_{\text{ideal}} = 1$$

$$CPI_{\text{new}} = (.7 * 1 + .3 * 1.5) = 1.15$$

Computer Organization

◆ Pipeline Systems

★ Data Hazard — Pipeline Scheduling or Instruction Scheduling

- Compiler attempts to schedule the pipeline to avoid the stalls by rearranging the code sequence to eliminate the hazard — **Software support to avoid data hazard.**
- Sometimes if compiler can not schedule the interlocks, a **no-op** instruction may be inserted.

Computer Organization

◆ Pipeline Systems

★ Data Hazard — Pipeline Scheduling or Instruction Scheduling

- Let us look at the following sequence of instructions:

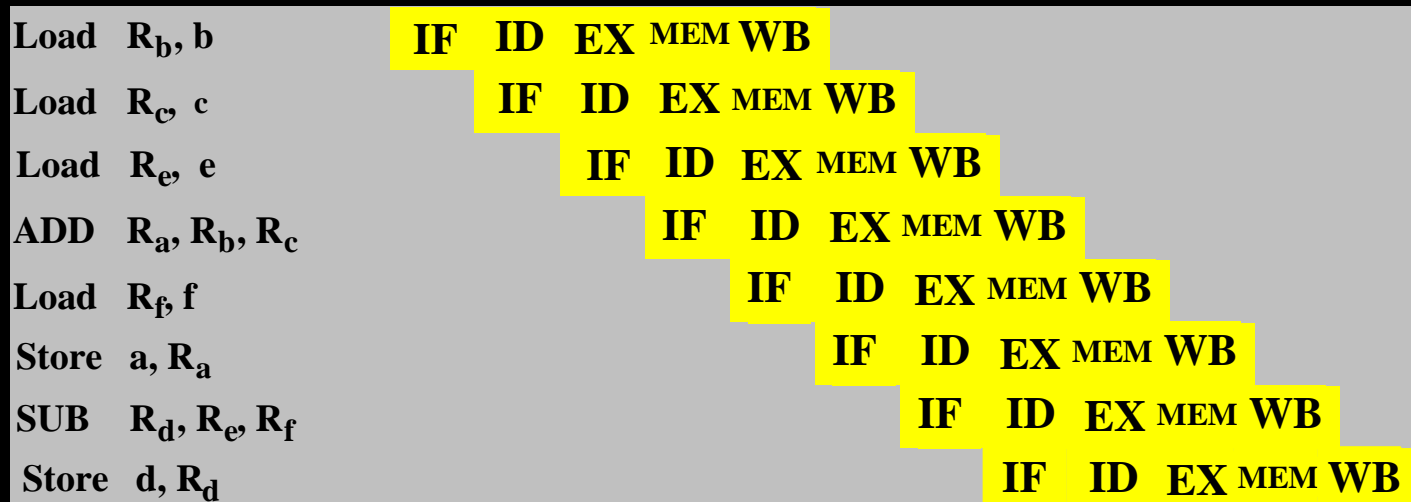
$$a = b + c$$

$$d = e - f$$

Computer Organization

◆ Pipeline Systems

★ Data Hazard — Pipeline Scheduling or Instruction Scheduling



Computer Organization



◆ Pipeline Systems

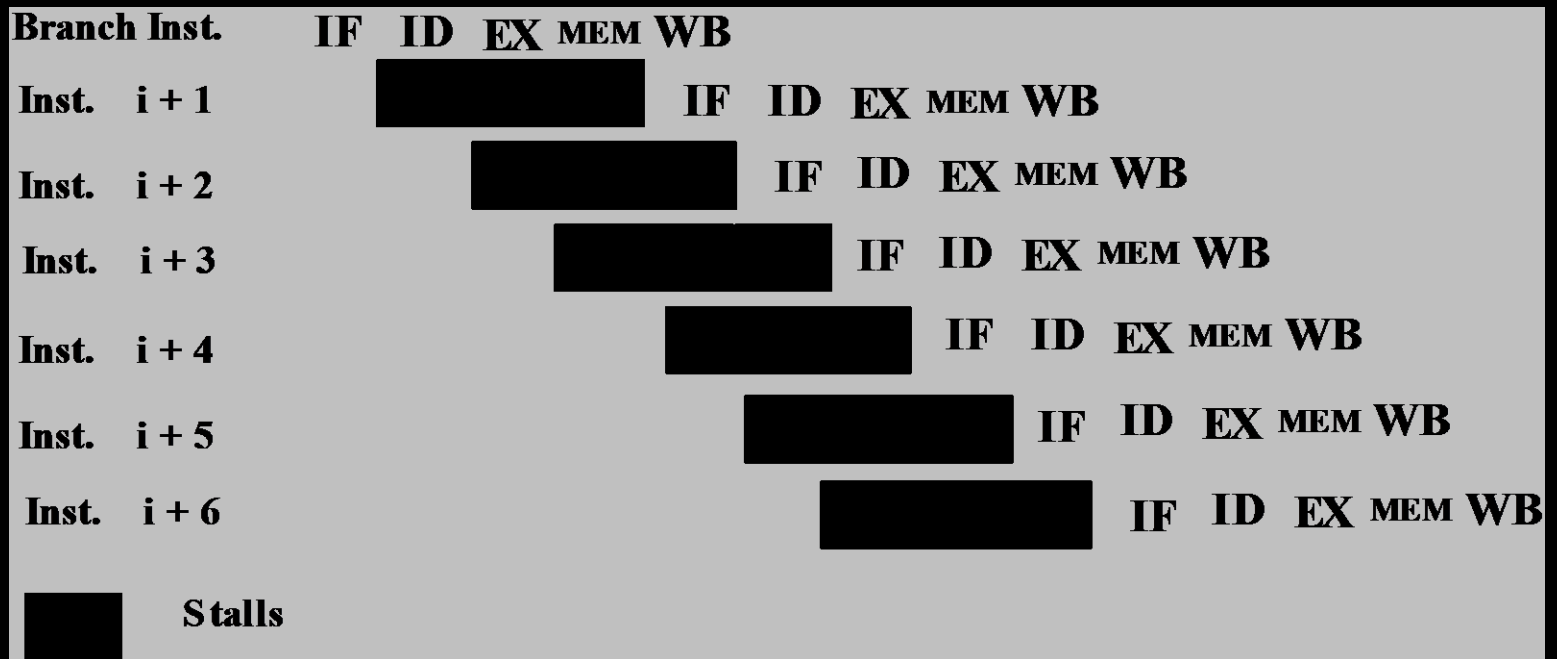
★ Control Hazard

- If instruction i is a successful branch, then the PC is changed at the end of MEM phase. This means stalling the next instructions for three clock cycles:

Computer Organization

◆ Pipeline Systems

★ Control Hazard



Computer Organization



◆ Pipeline Systems

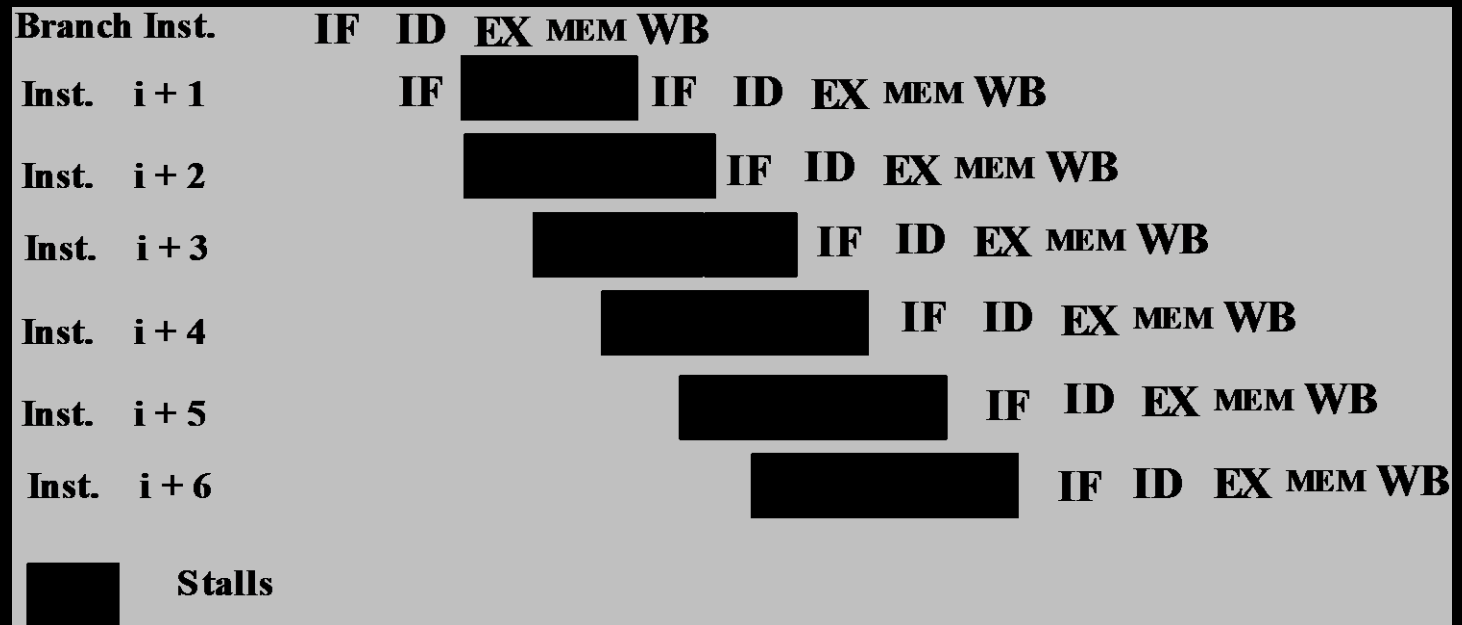
★ Control Hazard — Observations

- Three clock cycles are wasted for every branch.
- However, the above sequence is not even possible, since we do not know the nature of the instruction until after the instruction $i + 1$ is fetched.

Computer Organization

◆ Pipeline Systems

★ Control Hazard — Solution



Computer Organization



◆ Pipeline Systems

★ Control Hazard — Solution

- Still the performance penalty is severe.
- What are the solution(s) to speed up the pipeline?

Computer Organization

◆ Pipeline Systems

★ Control Hazard — Reducing pipeline branch penalties

- Detect, earlier in the pipeline, whether or not the branch is successful,
- For a successful branch, calculate the value of the *PC* earlier,
- It should be noted that, these solutions come at the expense of extra hardware,

Computer Organization

◆ Pipeline Systems

★ Control Hazard — Reducing pipeline branch penalties

- Freeze the pipeline — Holding any instruction after the branch until the branch destination is known — Easy to enforce,
- Assume unsuccessful branch — Continue to fetch instructions as if the branch were a normal instruction. If a branch is taken, then stop the pipeline and restart the fetch,

Computer Organization

◆ Pipeline Systems

★ Control Hazard — Reducing pipeline branch penalties

- Assume the branch is successful — as soon as the target address is calculated, fetch and execute instructions at the target,
- Delayed Branch — Software attempts to make the successor instruction valid and useful.

Computer Organization

◆ Pipeline Systems

★ Control Hazard

- **Delayed Branch:** Assuming branch is detected and calculated in *ID* stage.

Sequence of
instructions **before**

ADD R_1, R_2, R_3

IF $R_2 = 0$ then



Sequence of
instructions **after**

IF $R_2 = 0$ then

ADD R_1, R_2, R_3



Computer Organization

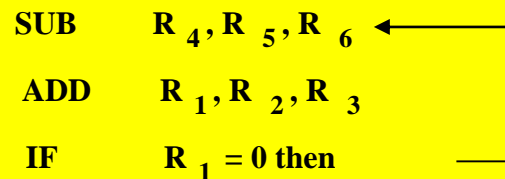
◆ Pipeline Systems

★ Control Hazard

- **Delayed Branch:** Assuming branch is detected and calculated in *ID* stage.

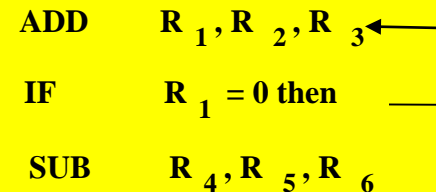
Sequence of
instructions before

SUB R₄, R₅, R₆ ←
ADD R₁, R₂, R₃
IF R₁ = 0 then



Sequence of
instructions after

ADD R₁, R₂, R₃ ←
IF R₁ = 0 then
SUB R₄, R₅, R₆



Computer Organization

◆ Pipeline Systems

★ Control Hazard

- **Delayed Branch:** Assuming branch is detected and calculated in *ID* stage.

