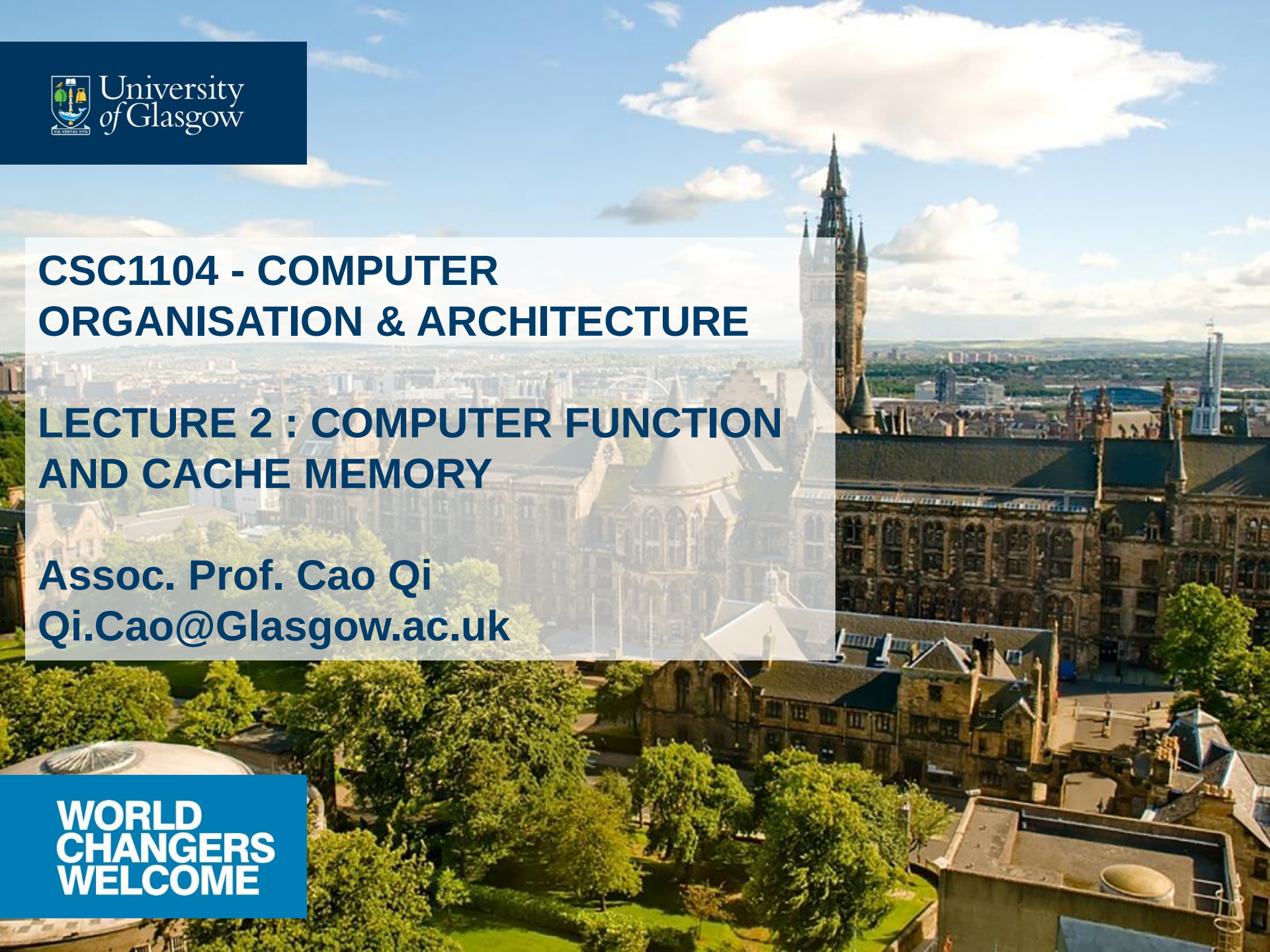


University of Glasgow

CSC1104 - COMPUTER ORGANISATION & ARCHITECTURE

LECTURE 2 : COMPUTER FUNCTION AND CACHE MEMORY

Assoc. Prof. Cao Qi
Qi.Cao@Glasgow.ac.uk

WORLD CHANGERS WELCOME

University of Glasgow

School of Computing Science

Acknowledgement

- Main contents of CSC1104 - Computer Organisation and Architecture are derived from:
  - **Computer organization and architecture, Designing for performance.** Author: William Stallings. Publisher: Pearson.

    Acknowledgement to: Author and Publisher.
  - **Computer organization and Design, The hardware/software interface.** Authors: D. Patterson and J. Hennessy. Publisher: Morgan Kaufmann.

    Acknowledgement to: Authors and Publisher.

University of Glasgow

School of Computing Science

Lecture Contents

- Computer Function:
  - **Instruction Fetch and Execute**
  - **Interrupts**
- Cache Memory:
  - **Characteristics of Memory Systems**
  - **Memory Hierarchy**
  - **Cache Memory Principles**

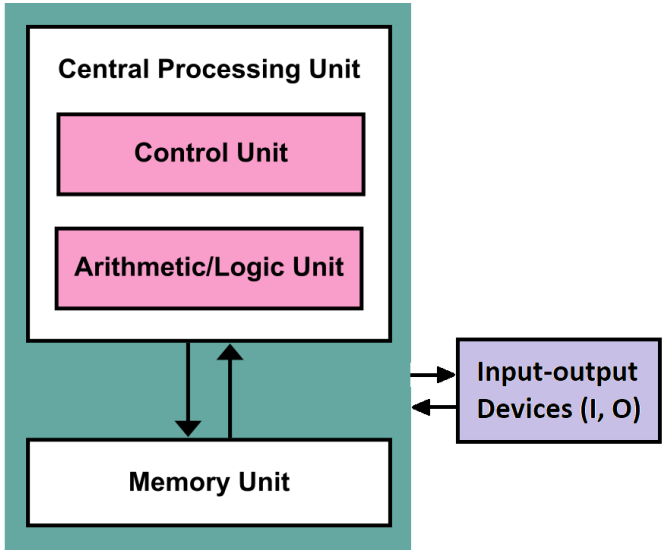
Computer Function

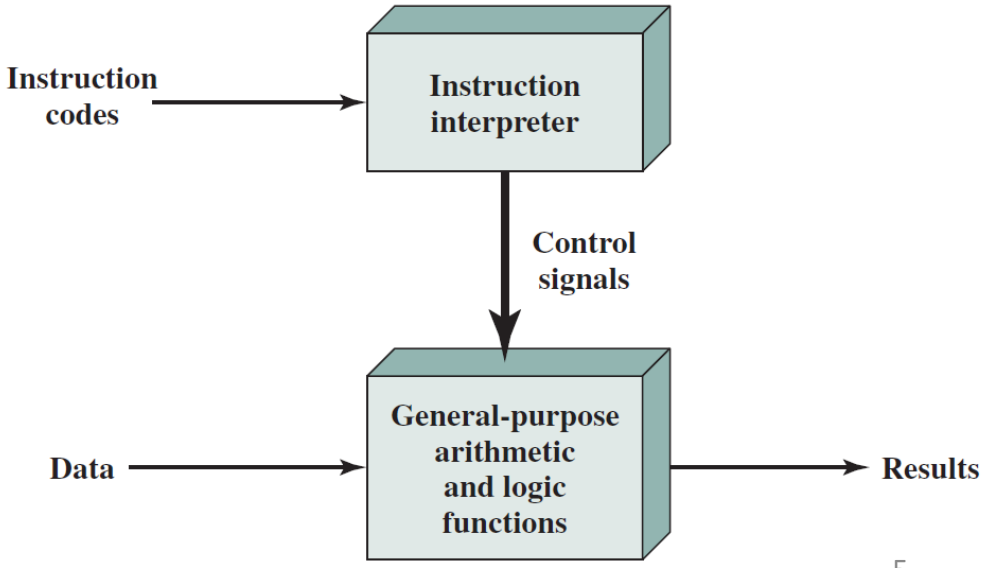
University of Glasgow

School of Computing Science

Von-Neumann Architecture (Recap)

- *Von-Neumann architecture* by John von Neumann.
  - **Data and instructions stored in the same memory.**
  - **Memory contents are accessible by addresses.**
  - **Instruction execution in a sequential way**

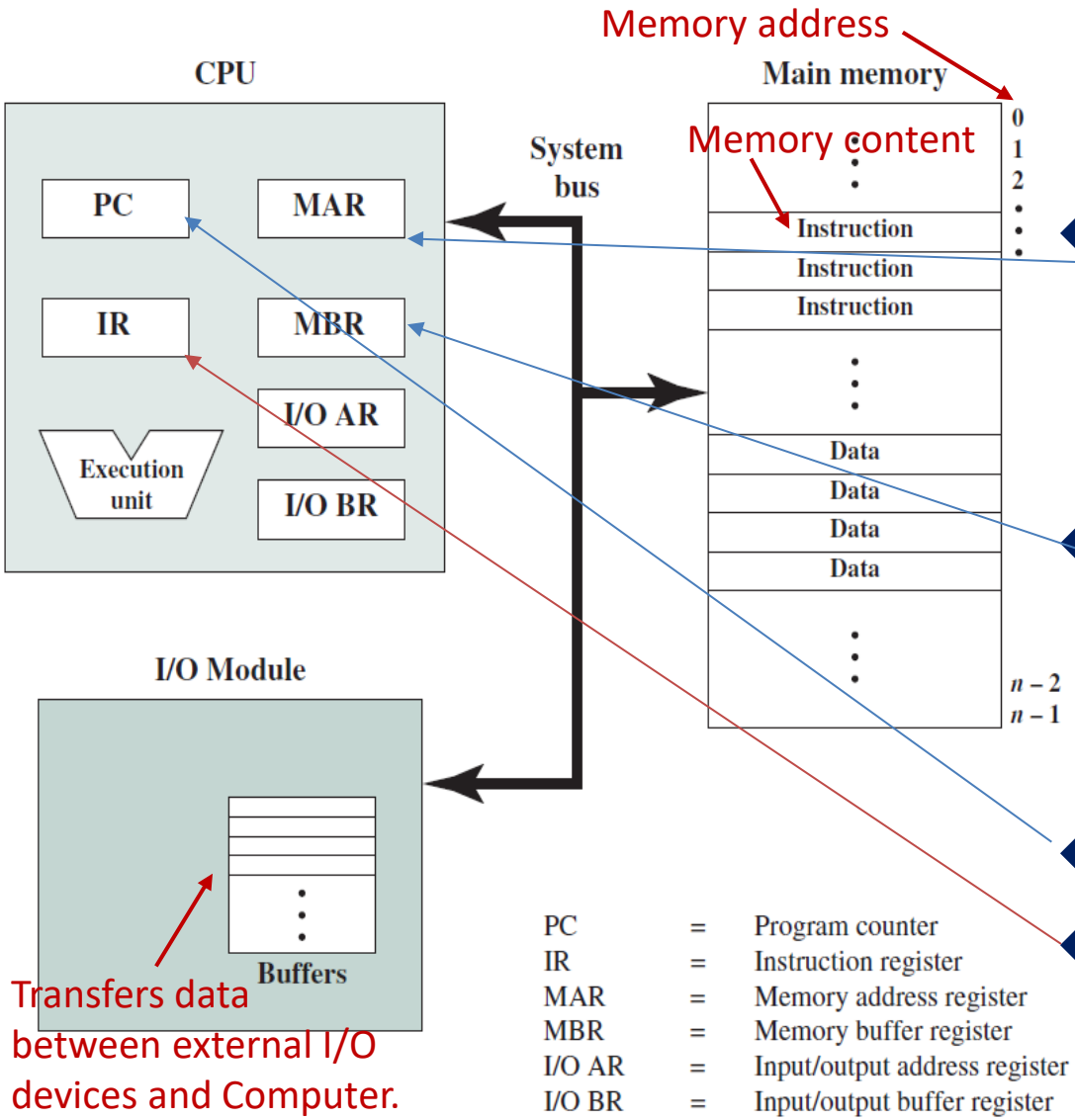




University of Glasgow

School of Computing Science

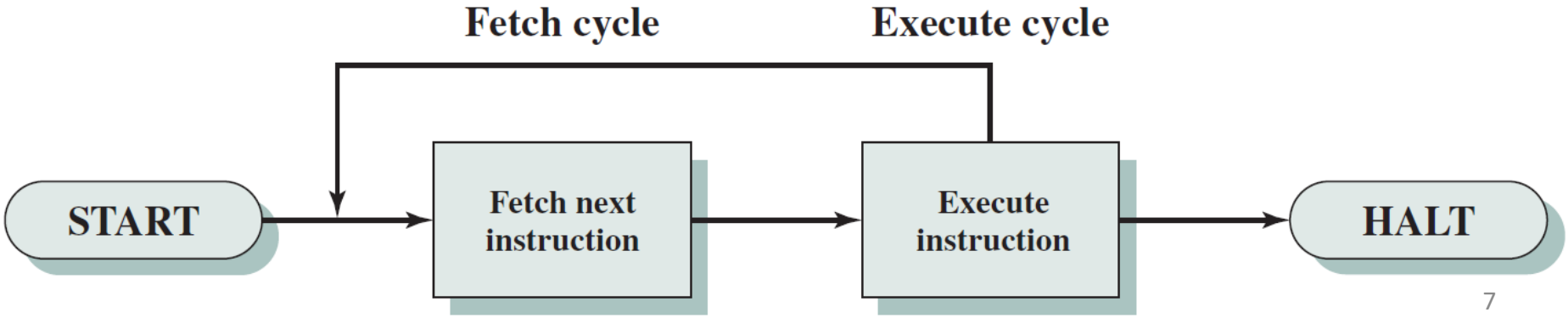
CPU Components: Top-Level View



- Memory address register (MAR):
  - specifies address in memory for the next read or write.
- Memory buffer register (MBR):
  - contains data read/write from/to memory.
- Program counter (PC):
- Instruction register (IR):

Computer Functions

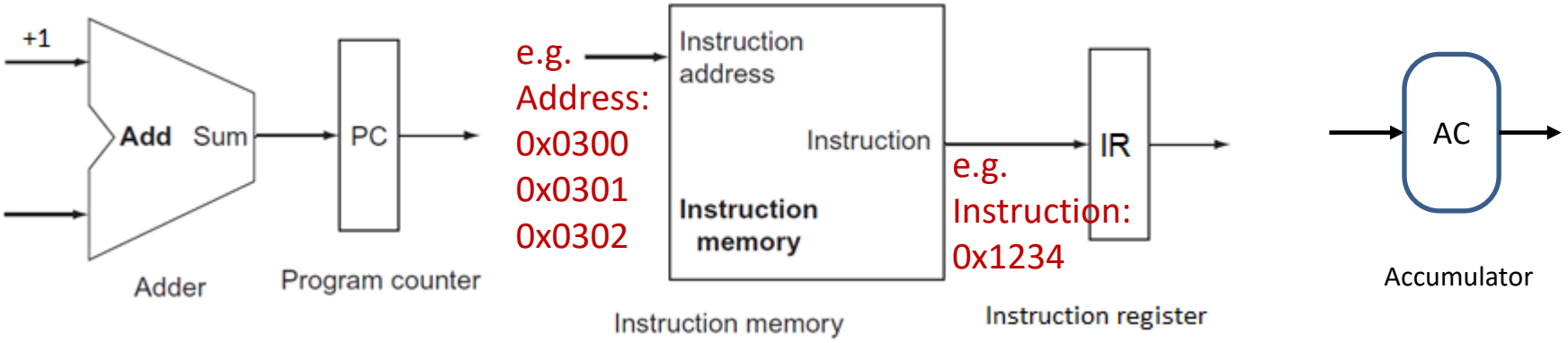
- Execute a program, with a set of instructions stored in memory.
- An instruction cycle: tasks required to process a single instruction:
  - **Fetch cycle:** processor reads instructions from memory one at a time.
  - **Execute cycle:** current instruction is executed.



University of Glasgow

School of Computing Science

Instruction Fetch and PC



- Processor fetches an instruction from memory to ***instruction register (IR)***.
- ***program counter (PC)*** register holds address of the next instruction to be fetched.
- PC is increased after each instruction fetch, pointing to the next instruction in sequence, unless special instructions received.
- ***Accumulator (AC)***: data register in CPU is for temporary storage.

Instruction Register (IR)

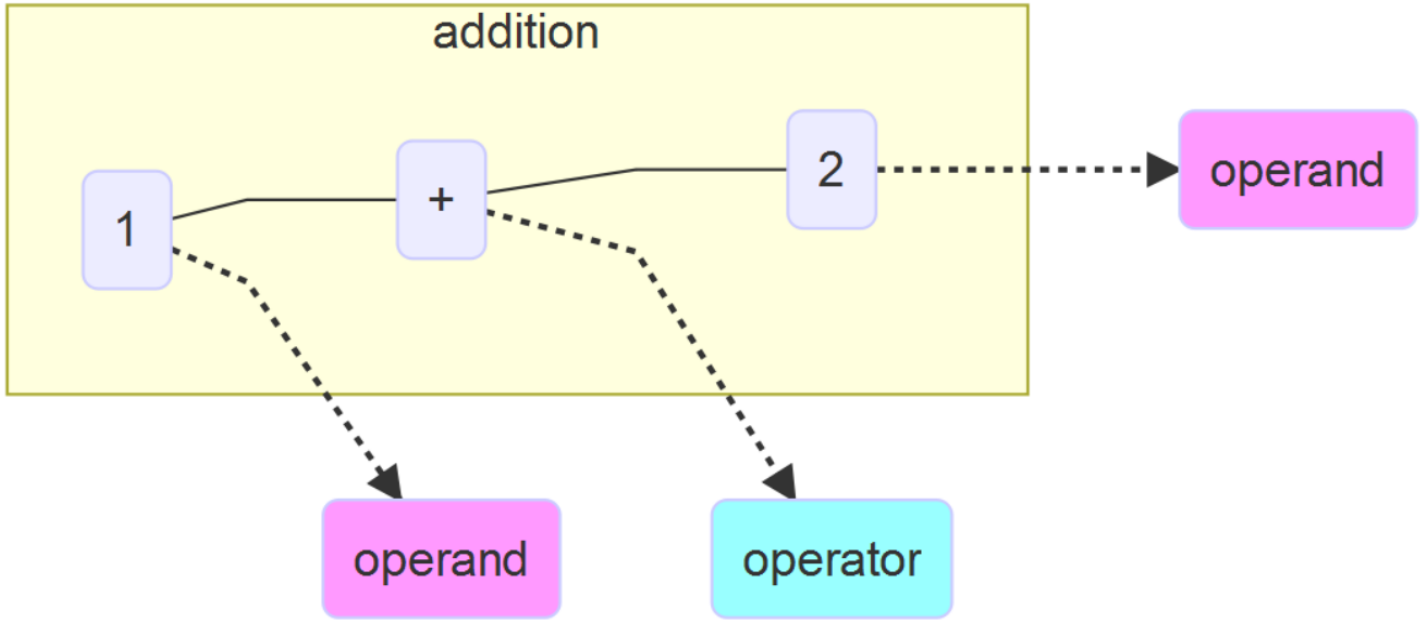
- Fetched instruction is loaded into the *IR*.
- Instruction specifies action to be taken, with 4 action categories:
  - **Processor-memory**: Data transferred in processor ⟺ memory.
  - **Processor-I/O**: Data transferred in processor ⟺ I/O peripherals.
  - **Data processing**: Processor performs certain operations on data.
  - **Control**: Alters sequence of instruction execution.
  - e.g., CPU fetches an instruction from memory 0x145, which is a control instruction to specify the next instruction being from memory 0x182.
  - CPU will then set the program counter (PC) to 0x182.
  - On the next fetch cycle, instruction fetched from memory 0x182, rather than 0x146 (i.e., 0x145 + 1 = 0x146).

University of Glasgow

School of Computing Science

Operators, operands, operations

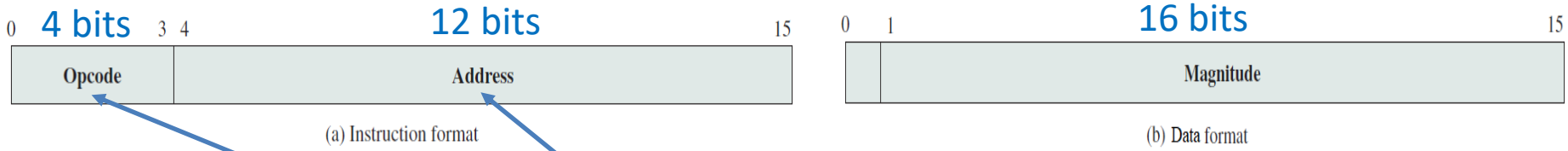
- Operators are expressions to do some operation (e.g., add, subtract, multiply, store).
- Operands are the values the operators do operations upon.



University of Glasgow

School of Computing Science

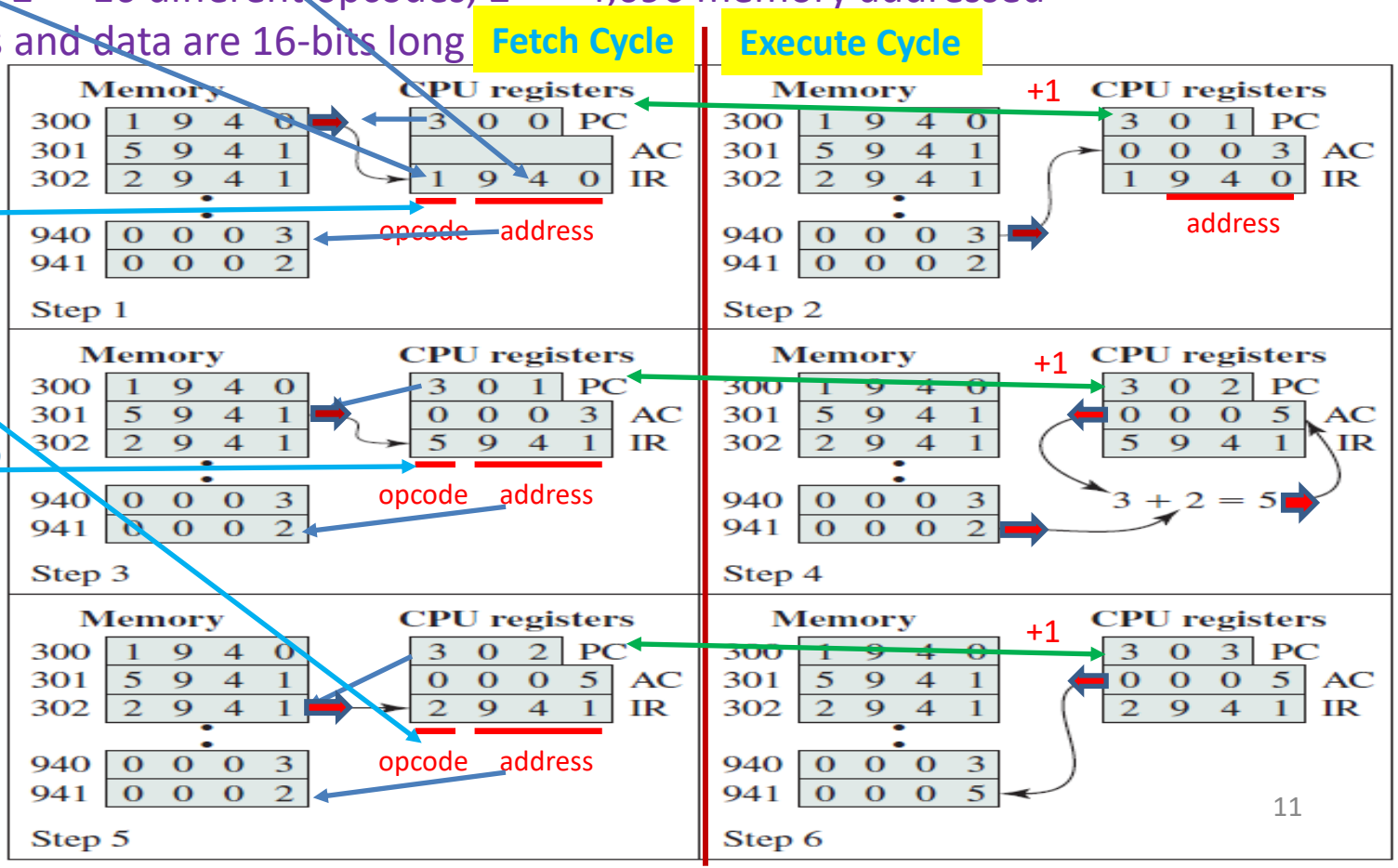
Example 2.1 - A Partial Program Execution



- Instruction code: $2^4 = 16$ different opcodes, $2^{12} = 4{,}096$ memory addressed
- Both instructions and data are 16-bits long

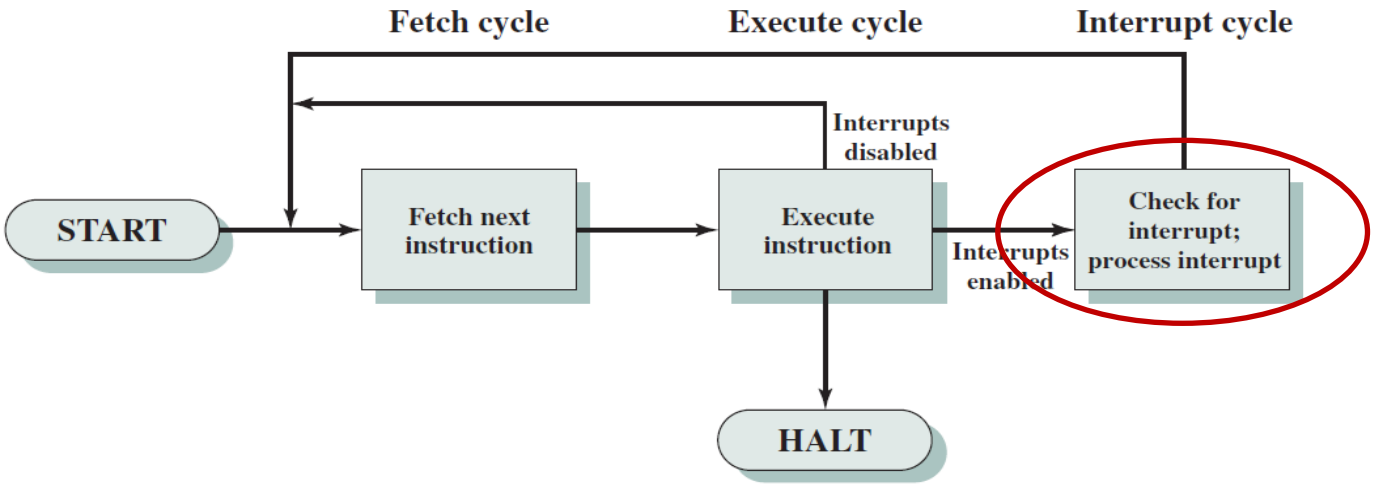
Example opcodes:

- 0x1 = Load AC from memory
- 0x2 = Store AC to memory
- 0x5 = Add data from memory into AC



Interrupts

- Almost all computers provide a mechanism by other modules (I/O, memory, etc.) to interrupt normal processing of CPU.
- Interrupts as a way to improve processing efficiency. Most external devices are much slower than CPU.



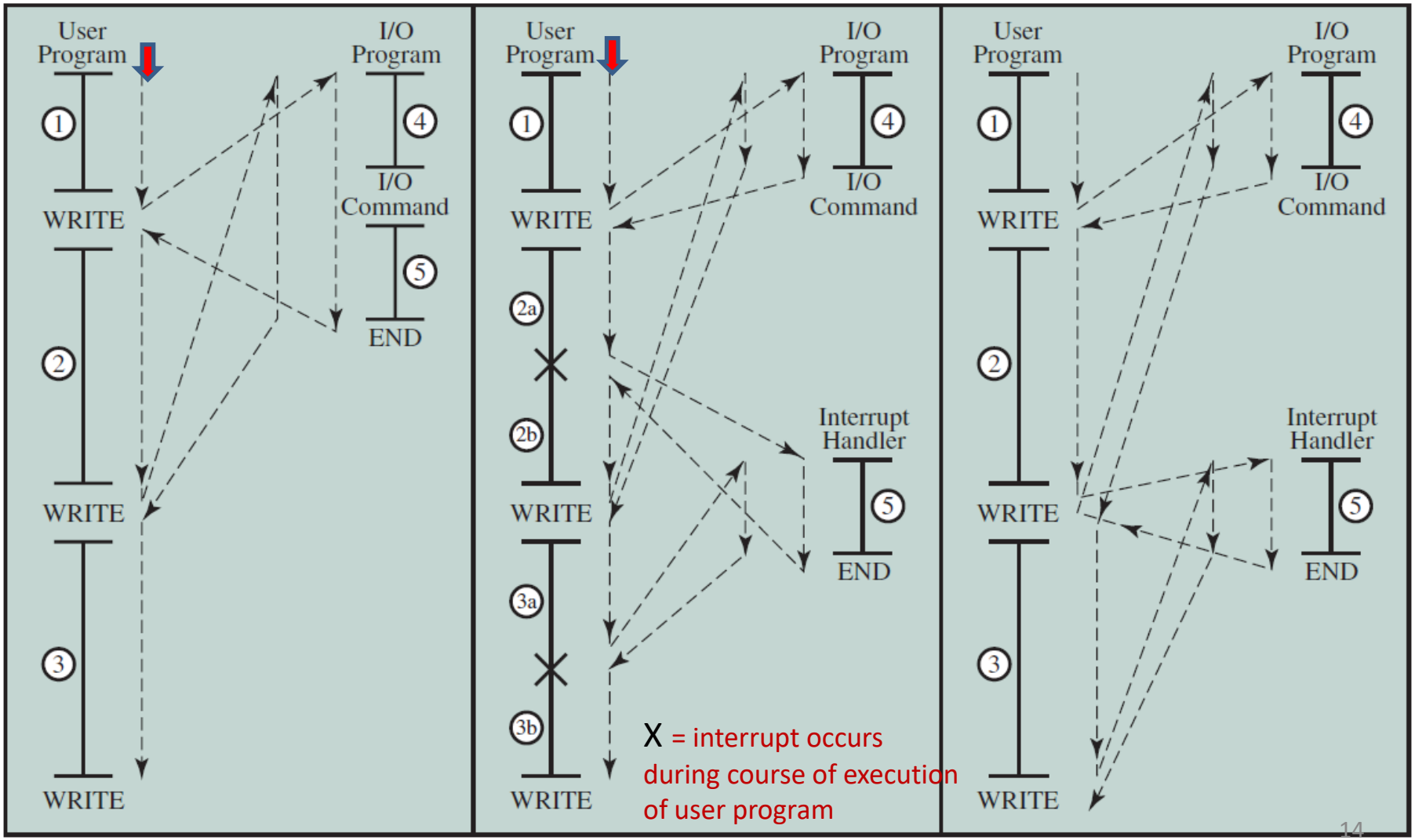
Classes of Interrupts

Classes of Interrupts	Descriptions
Program	Result of arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by timer within CPU. It allows operating system to perform certain functions on a regular basis.
I/O	Generated by I/O controller, to signal normal completion of an operation, request service from CPU, or to signal a variety of error conditions.
Hardware Failure	Generated by a failure such as power failure or memory parity error.

University of Glasgow

School of Computing Science

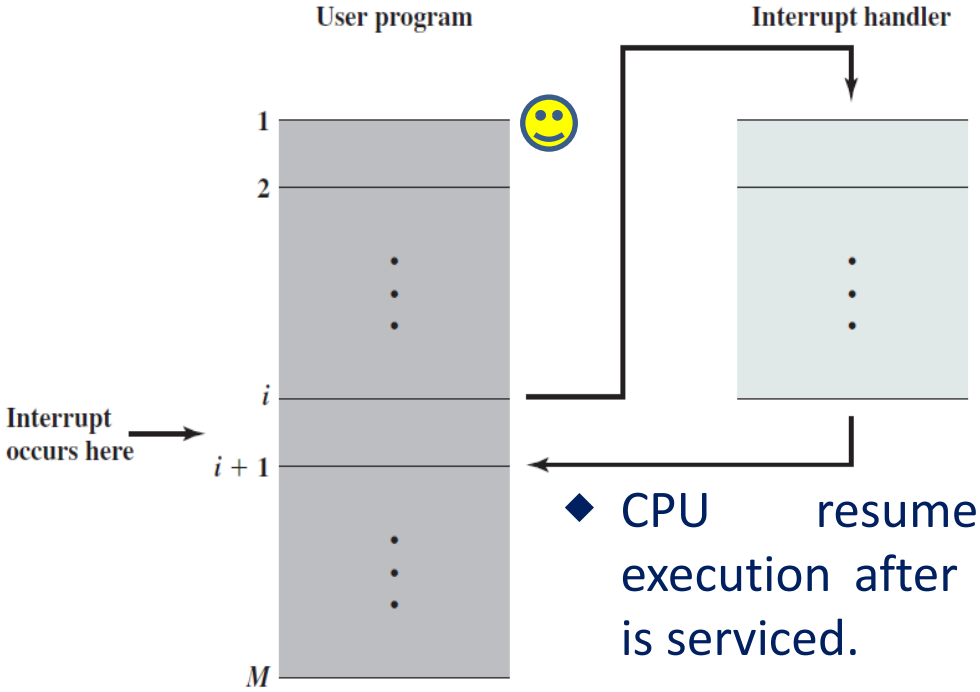
Program Flow without and with Interrupts



(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

Interrupt Cycle

- Receiving ***interrupt request***, CPU checks which interrupt occurs.
- For a pending interrupt, ***interrupt handler*** is performed:
  - CPU suspends execution of the current program; saves its context (program counter; data relevant to current activity).
  - Sets program counter (PC) to the starting address of an ***interrupt handler***.



- CPU resumes original execution after I/O interrupt is serviced.

University of Glasgow

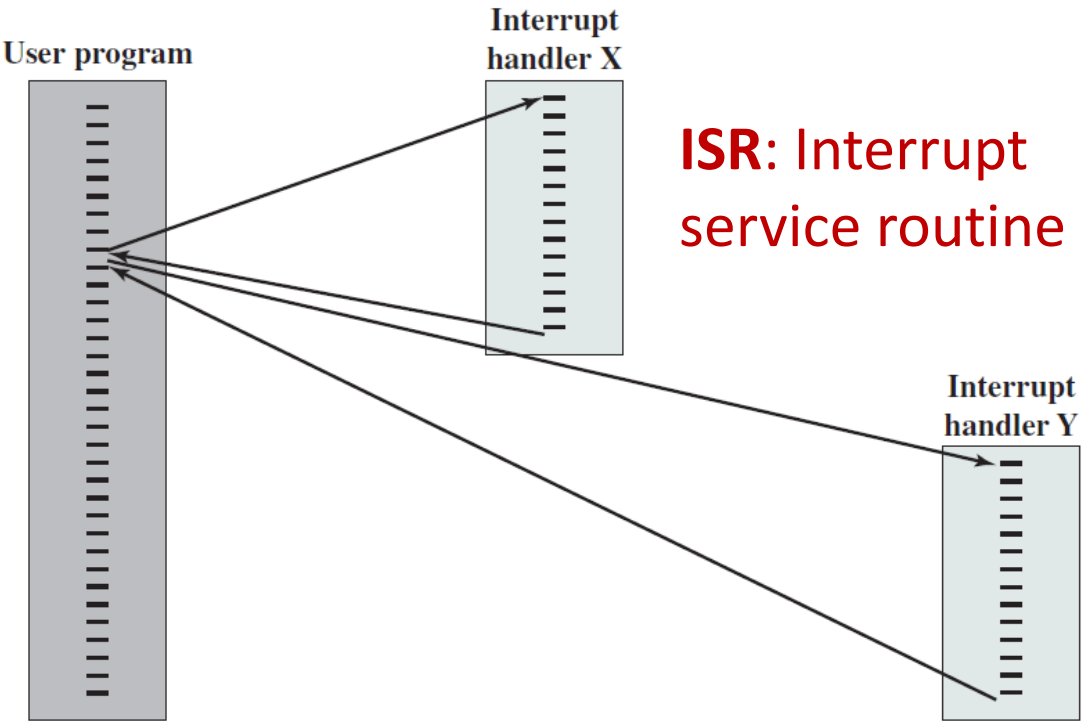
School of Computing Science

Sequential Interrupt Approach

1. Disable interrupts while an interrupt is being processed.

   when an interrupt occurs, CPU interrupts are disabled immediately. New interrupt requests won't be responded.

- Interrupts are handled in sequential order.
- After ISR completes, CPU interrupts are enabled. Before resuming user program, CPU checks if any interrupts occur but yet to respond.



(a) Sequential interrupt processing

ISR: Interrupt service routine

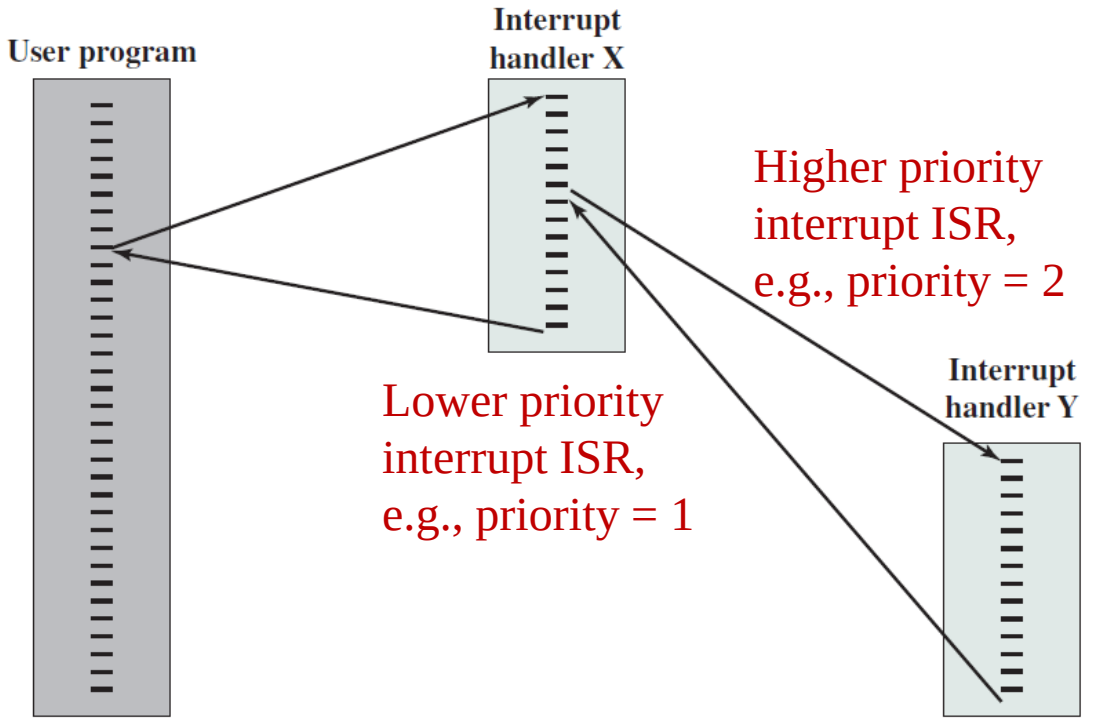
University of Glasgow

School of Computing Science

Nested Interrupt Approach

2. Define priorities for interrupts. Allow higher priority interrupt to be serviced first. It can interrupt lower-priority *interrupt service routine (ISR)*.

- Devices are assigned by different priorities of interrupt handler.
- The higher the number, the higher the interrupt priority.
- Interrupt from higher priority devices are responded first by CPU.

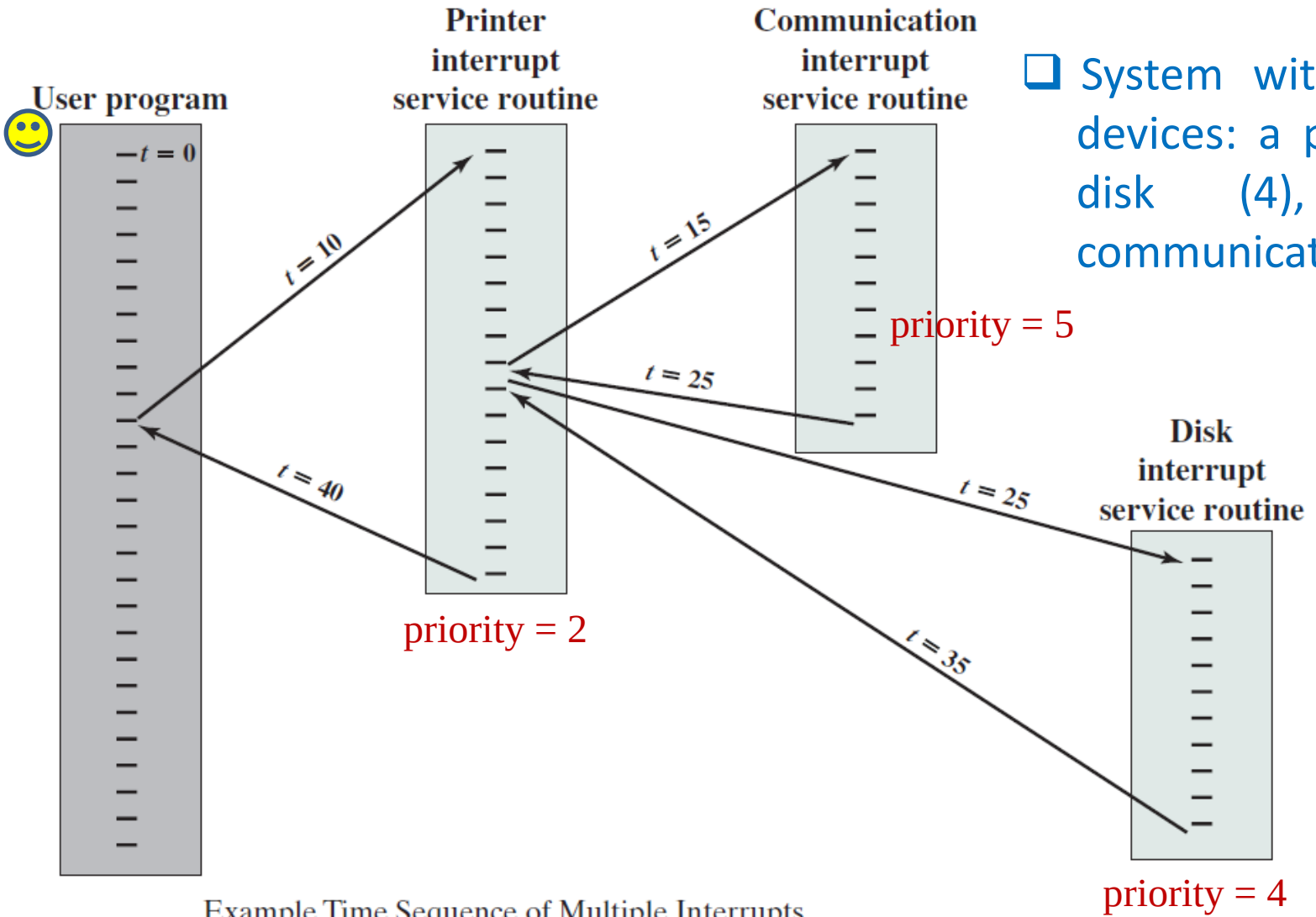


(b) Nested interrupt processing

University of Glasgow

School of Computing Science

Example 2.2 – ISR of Multiple Interrupts



Example Time Sequence of Multiple Interrupts

- System with three I/O devices: a printer (2), a disk (4), and a communications line (5).

Memory Hierarchy and Cache Memory

University of Glasgow

School of Computing Science

Important Characteristics of Memory

1. Capacity of memory: (bytes or words) 1 word = 2 bytes; 1 byte = 8 bits.
2. Performance of memory
   - **Access time (latency):** time from an address is presented to the memory, to data stored or read out.
   - **Memory cycle time**: access time + any additional time required before the next access can commence.
   - **Transfer rate**: the rate at which data can be transferred into or out of a memory unit.
     - For random-access memory: 1/(clock cycle time).
     - For non-random-access memory: $T_n = T_A + \frac{n}{R}$

       where T_n = Average time to read or write n bits.

       T_A = Average access time.

       n = Number of bits to read/write.

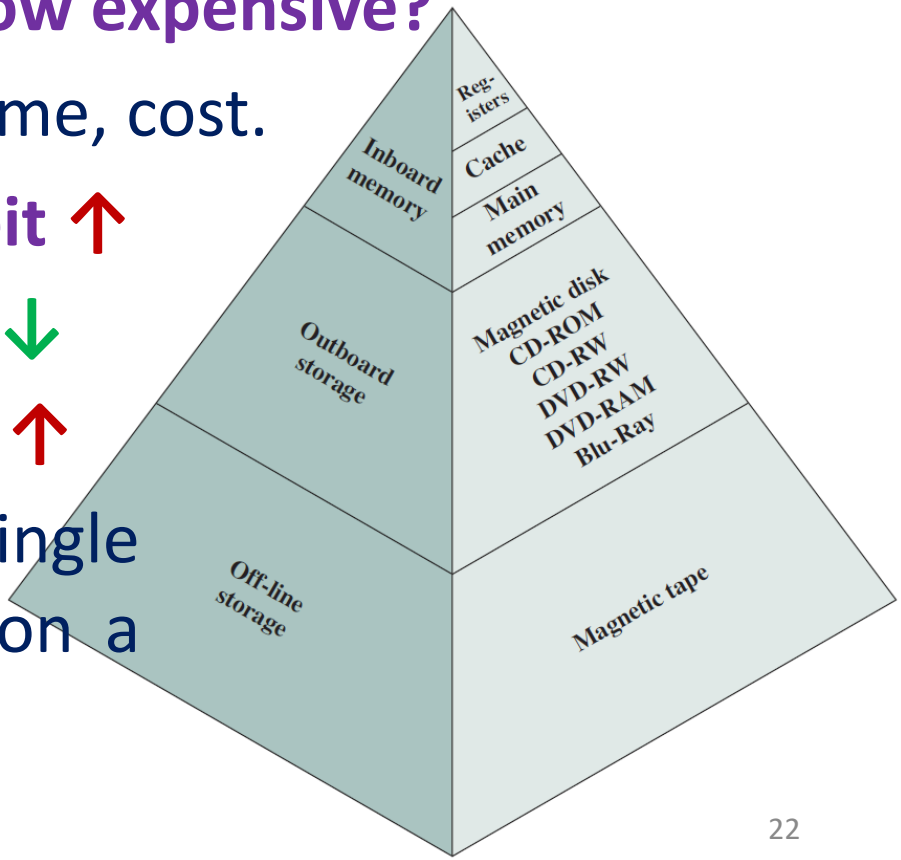
       R = Transfer rate, in bits per second (bps)

Physical Characteristics

- Volatile memory: data decayed naturally or lost when electrical power is off.
- Nonvolatile memory: no electrical power needed to retain data once recorded.
- Nonerasable memory: cannot be altered, unless destroying storage units, i.e., Read-only memory (ROM).
- Erasable memory: can be altered and erased from storage.
- Semiconductor memory: either volatile or nonvolatile.
- Magnetic-surface memories: nonvolatile

Memory Hierarchy

- Design constraints on a computer's memory:
  - **how large? how fast? how expensive?**
- Trade-off: capacity, access time, cost.
  - **Access time ↓ cost per bit ↑**
  - **Capacity ↑ cost per bit ↓**
  - **Capacity ↑ access time ↑**
- Solution: not to rely on single memory components, but on a memory hierarchy.

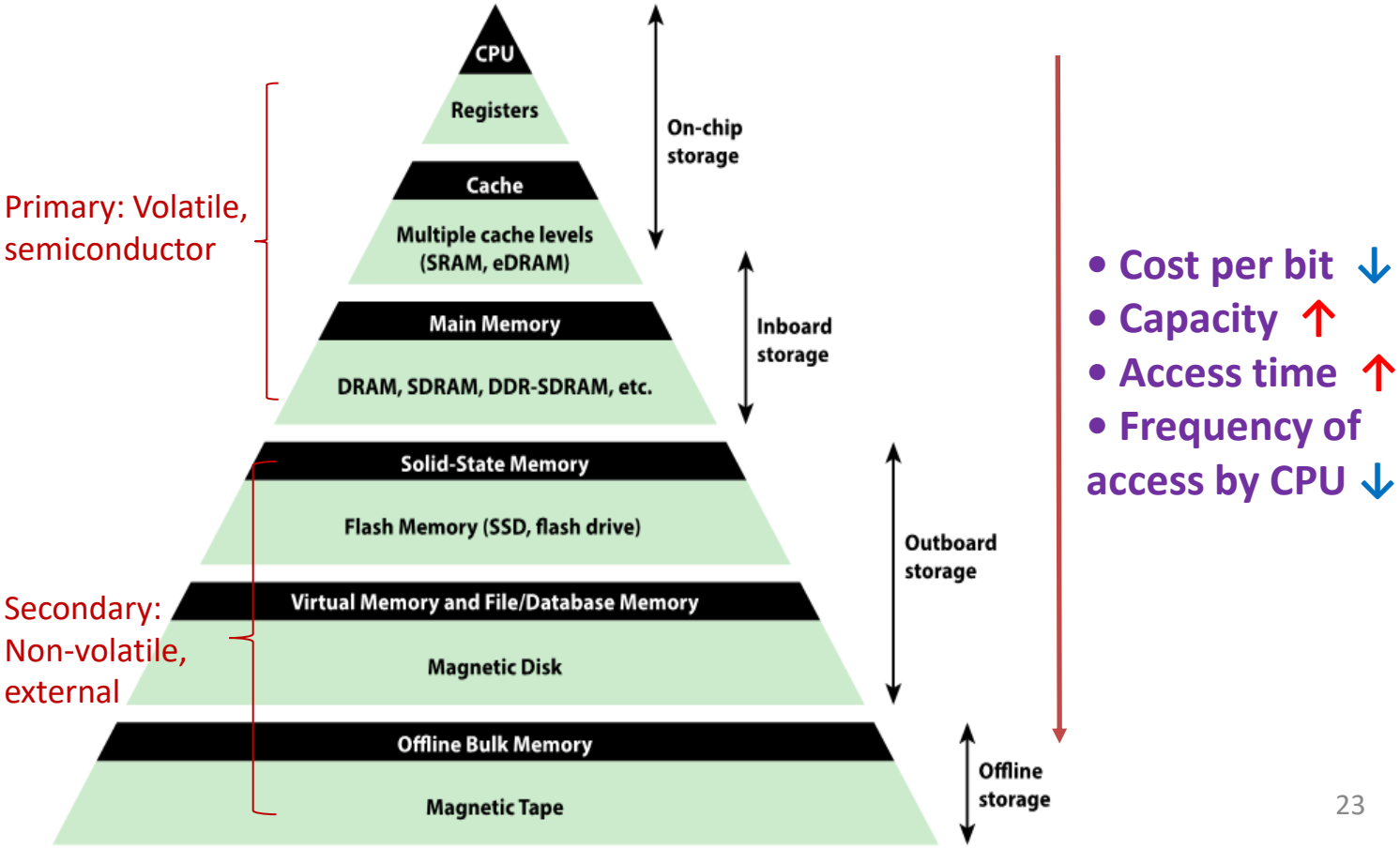


University of Glasgow

School of Computing Science

Relative Cost, Size and Speed Characteristics

- Smaller, more expensive, faster memories supplemented by larger, cheaper, slower memories.



University of Glasgow

School of Computing Science

Performance of Accesses 2 Levels of Memory

- Two-level memory: M1 is smaller, faster, more expensive than M2.
- If data is in Level M1, CPU can access directly. If in Level M2, data is first transferred to Level M1, then accessed by CPU.
  - M2 contain all program instructions and data.
  - Most recently accessed instructions and data are M1.
  - Cluster data in M1 need be swapped back to M2 regularly.
- Hit ratio *H*: probability of data is found in M1.
  - T_1: access time to M1
  - T_2: access time to M2
  - Total access time per word = $H \times T_1 + (1 - H) \times (T_1 + T_2)$

University of Glasgow

School of Computing Science

Example 2.3 – Accesses 2 Levels of Memory

- For a two levels of memory system, Level M1 contains 1000 bytes and has an access time of 0.01 *μs*; Level M2 contains 100,000 bytes with an access time of 0.1 *μs*. The hit ratio is 95%. Calculate the average time to access a word by the CPU?

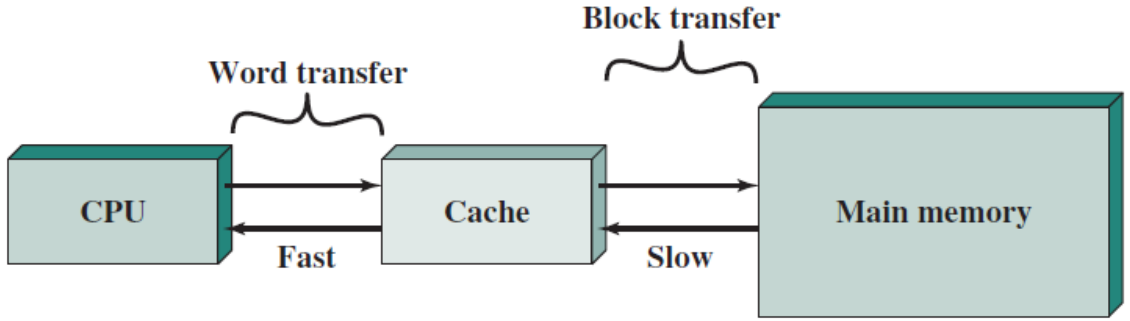
Solution:

a) $T_1 = 0.01$ μs. $T_2 = 0.1$ μs. $H = 0.95$

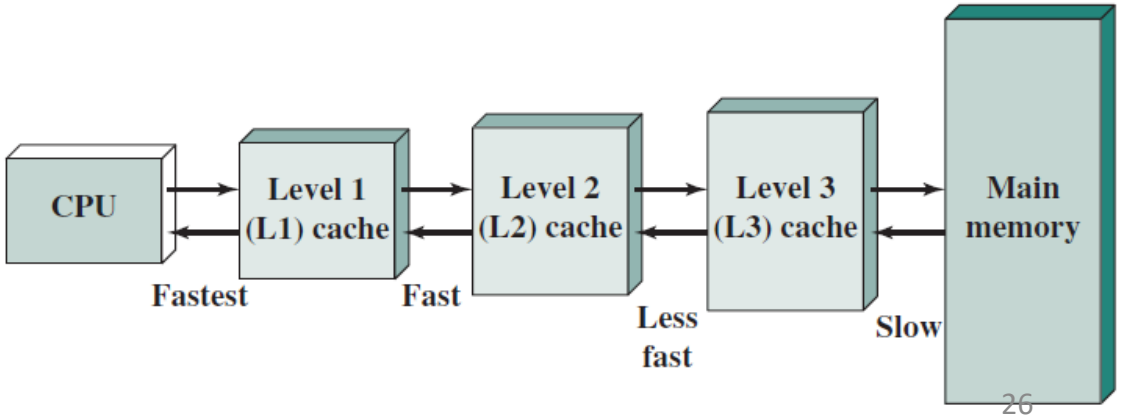
b) Time = $H \times T_1 + (1 - H) \times (T_1 + T_2)$ = 0.95×0.01 μs + $0.05 \times (0.01$ μs + 0.1 μs) = 0.0095 + 0.0055 = 0.015 μs.

Cache Memory Principles

- When CPU attempts to read a word from memory address, checks if it is in cache. If so, delivered to CPU directly.
- If not, a data block of memory read into cache, word is delivered to CPU.
- **Locality of reference**: when a data block fetched into cache from memory, likely there will be future references to data in the same block.



(a) Single cache

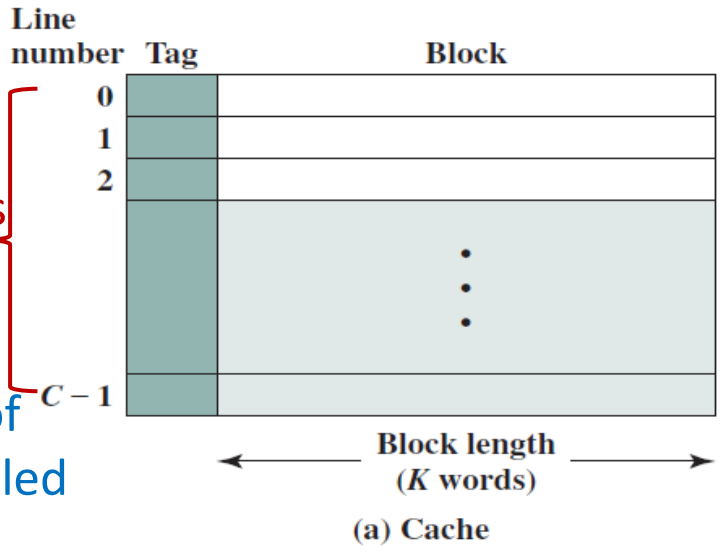


(b) Three-level cache organization

University of Glasgow

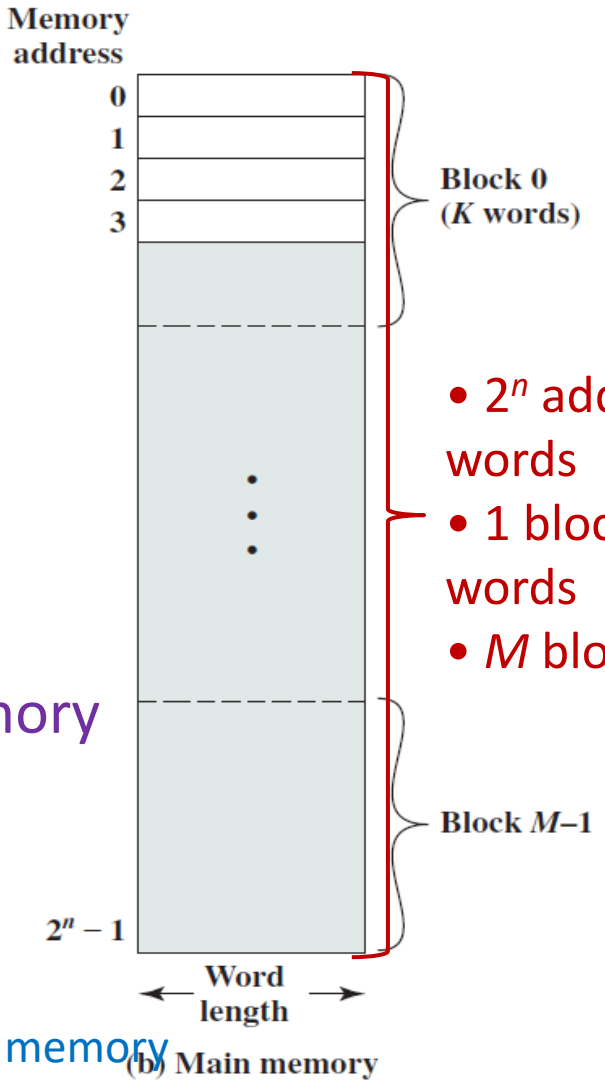
School of Computing Science

Cache/Main Memory Structure



C blocks (lines)

Cache consists of blocks called lines.



- 2^n addressable words
- 1 block = *K* words
- *M* blocks

- 1 line = *K* words + tag (+ control bits)
- Size of 1 cache line = size of 1 memory block = *K* words
- No. of blocks in memory: $M = 2^n/K$
- $C << M$

1 block = minimum unit of transfer between cache and memory

Elements of Cache Design

- Basic design elements to classify and differentiate cache architectures.

Cache Addresses
- Logical
- Physical

Cache Size 

Mapping Function
- Direct 
- Associative
- Set associative

Replacement Algorithm
- Least recently used (LRU)
- First in first out (FIFO)
- Least frequently used (LFU)
- Random

Write Policy
- Write through
- Write back

Line Size

Number of Caches
- Single or two level
- Unified or split

Cache Capacity (Cache Size)

- Preferable cache capacity:
  - Small enough: average cost per bit ≈ main memory alone.
  - Large enough: average access time ≈ small cache alone.
- Motivations for minimizing cache capacity:
  - Cache capacity ↑, number of gates ↑, number of address bits ↑
    => Access time of larger cache ↑
  - Chips and circuit boards area limits cache capacity.
- Cache performance is very sensitive to the nature of workload, impossible to get a single "optimum" cache capacity.

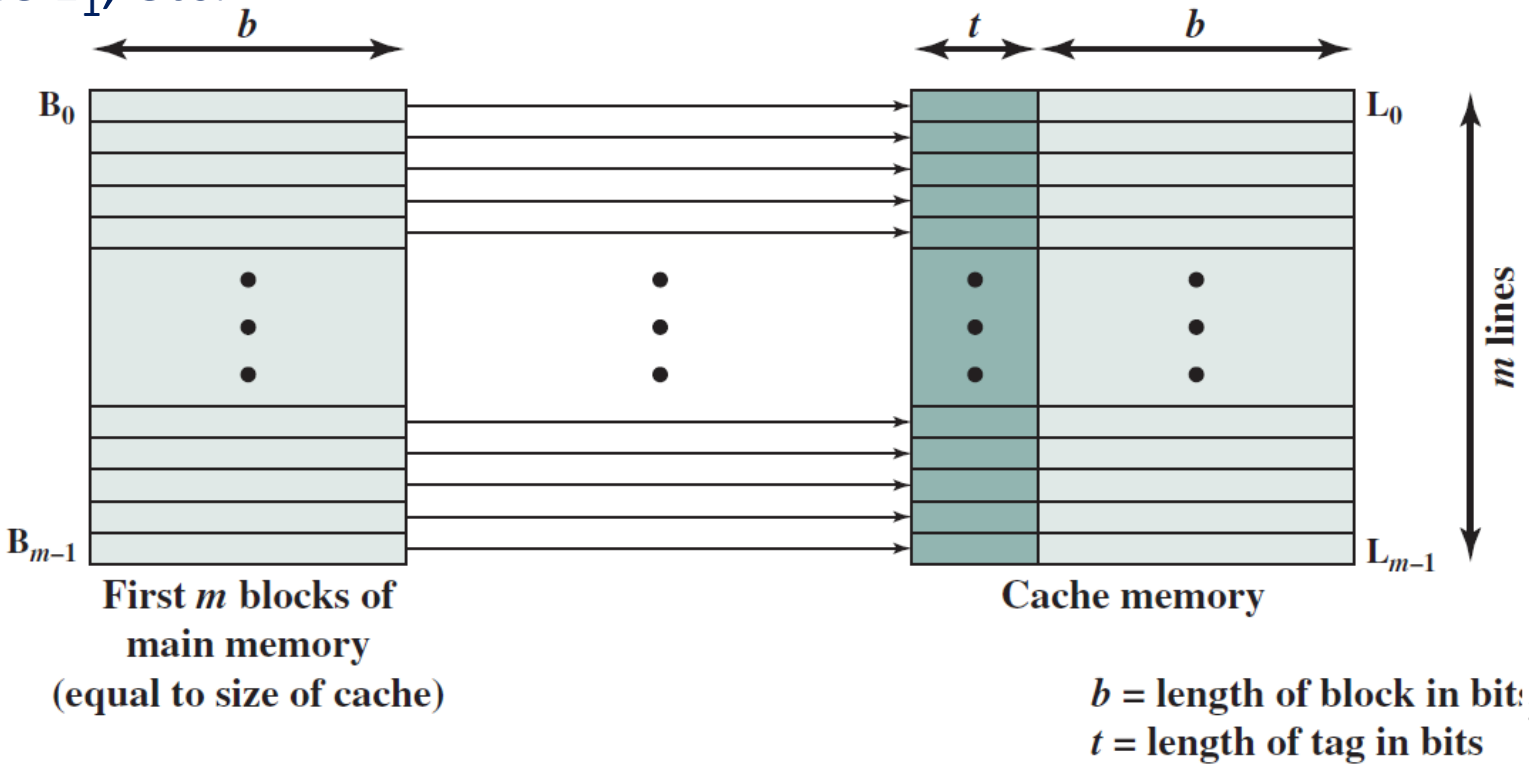
Mapping Function and Cache Access Methods

- No. of cache lines << No. memory blocks, an algorithm needed for mapping main memory blocks into cache lines.

Method	Organization	Mapping of Memory to Cache	Access using Main Memory Address
Direct Mapped	Sequence of *m* lines	Each block of main memory maps to a unique cache line.	***Line*** of address used to access cache line; ***Tag*** bits used to check for *hit on that line*.

Direct Mapped Cache

- Maps the first *m* blocks of memory, each block into only one cache line.
- Next *m* blocks of main memory map into cache in same way. i.e., block B_m maps into line L_0 of cache, block B_m+1 maps into line L_1, etc.



Features of Direct Mapped Technique

- Blocks of main memory are assigned to cache lines:

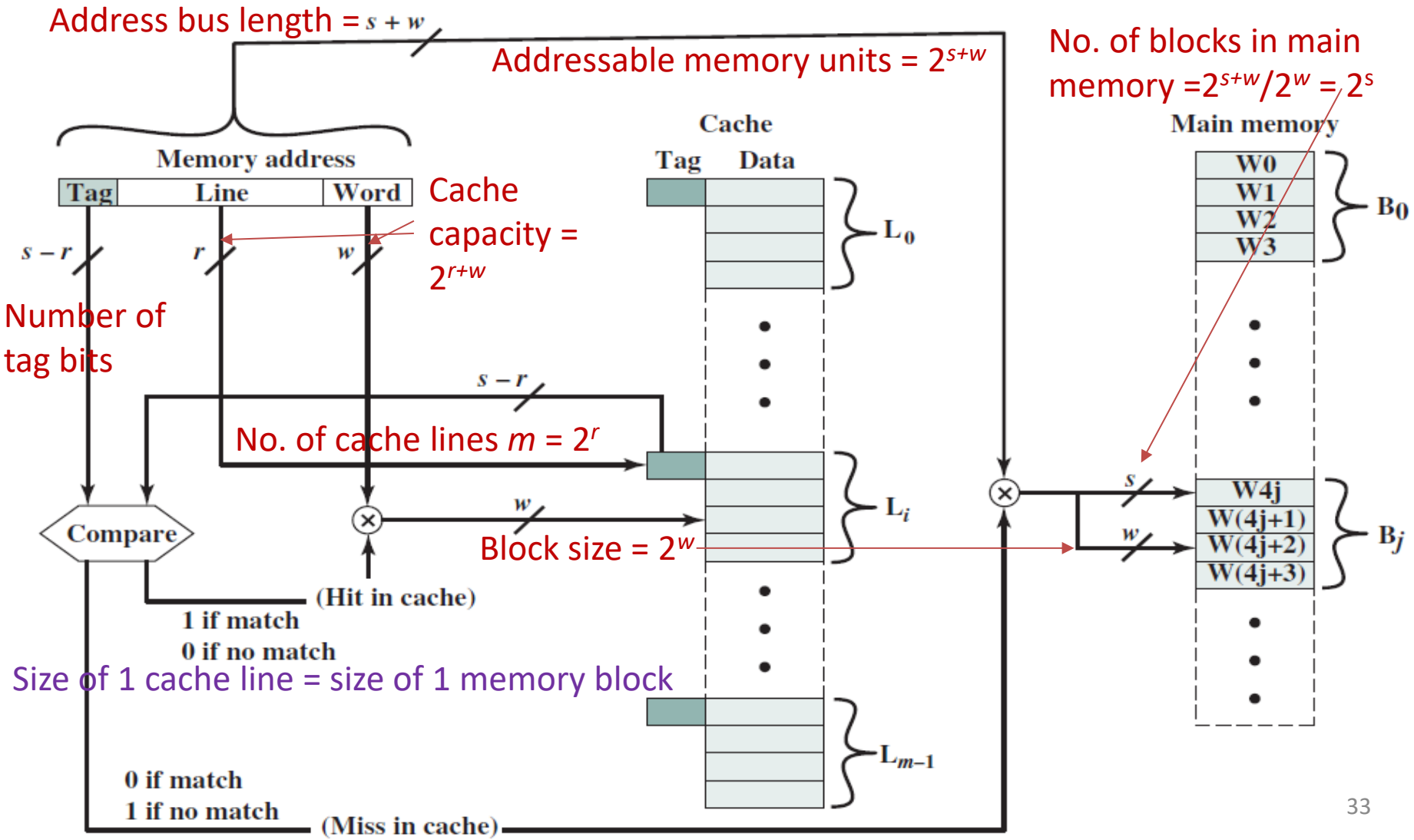
Cache line	Main memory blocks assigned
0	$0, m, 2m, \ldots, 2^s - m$
1	$1, m + 1, 2m + 1, \ldots, 2^s - m + 1$
⋮	⋮
$m - 1$	$m - 1, 2m - 1, 3m - 1, \ldots, 2^s - 1$

- Pros: simple and inexpensive to implement.
- Cons: a fixed cache line location for any given block.
- ***Thrashing:*** If CPU references words repeatedly from 2 different blocks mapping into same cache line, the blocks keep being swapped in cache. Hit ratio is low.

University of Glasgow

School of Computing Science

Direct Mapped Cache Organization



University of Glasgow

School of Computing Science

Example 2.4 – Direct Mapped Cache

- A memory with 32-bit address (1 byte per address unit), with the number of cache line as 64. Direct mapped cache with number of tag bits are 20 bits. Calculate following parameters: No. of addressable memory units, No. of blocks in main memory, and the cache capacity.

Solution:

- For Direct Mapped Cache, No. of tag bits = ($s - r$) bits; r is No. of address bits for line numbers, w is No. of address bits for a block.
- Memory address length is 32-bit. Hence $s + w = 32$.
- No. of addressable memory units = $2^{s+w} = 2^{32}$ bytes.
- No. of tag bits: $s - r = 20$ bits;
- Number of cache line is 64: $m = 2^r = 64$; ⇒ $r = 6$ bits.
- Hence: $s = 20 + 6 = 26$ bits. $w = 32 - 26 = 6$ bits.
- No of blocks in main memory= $2^s = 2^{26}$.
- Cache capacity = $2^{r+w} = 2^{6+6} = 2^{12}$ bytes.

Next Lecture

Lecture 3: Internal Memory and External Memory