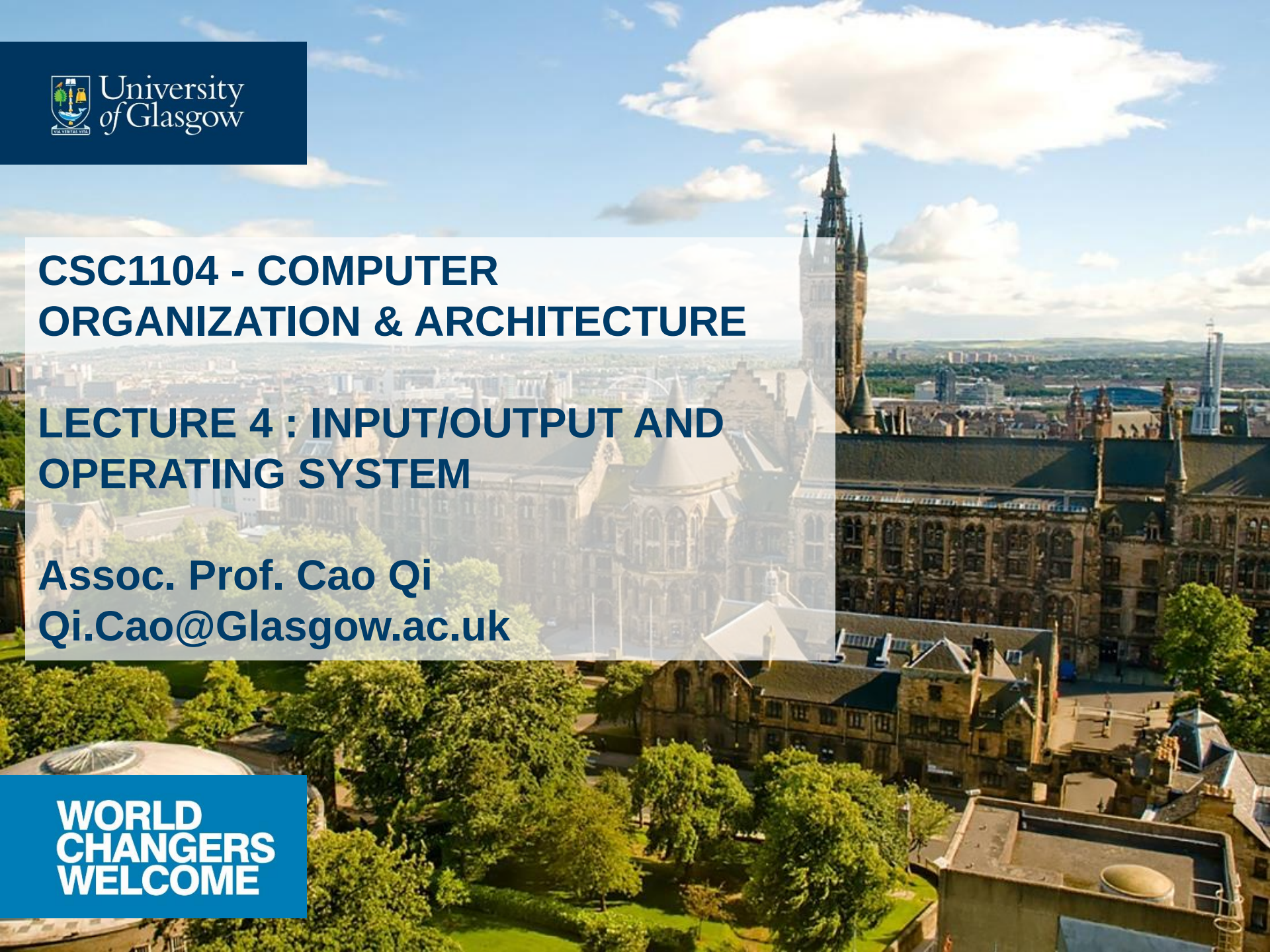


University of Glasgow

CSC1104 - COMPUTER ORGANIZATION & ARCHITECTURE

LECTURE 4 : INPUT/OUTPUT AND OPERATING SYSTEM

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.

Lecture Contents

- Input/Output:
  - **I/O Module**
  - **I/O Operations**
  - **Direct Memory Access (DMA)**
- Operating System (OS):
  - **Process Model**
  - **Memory Management**

Input Output

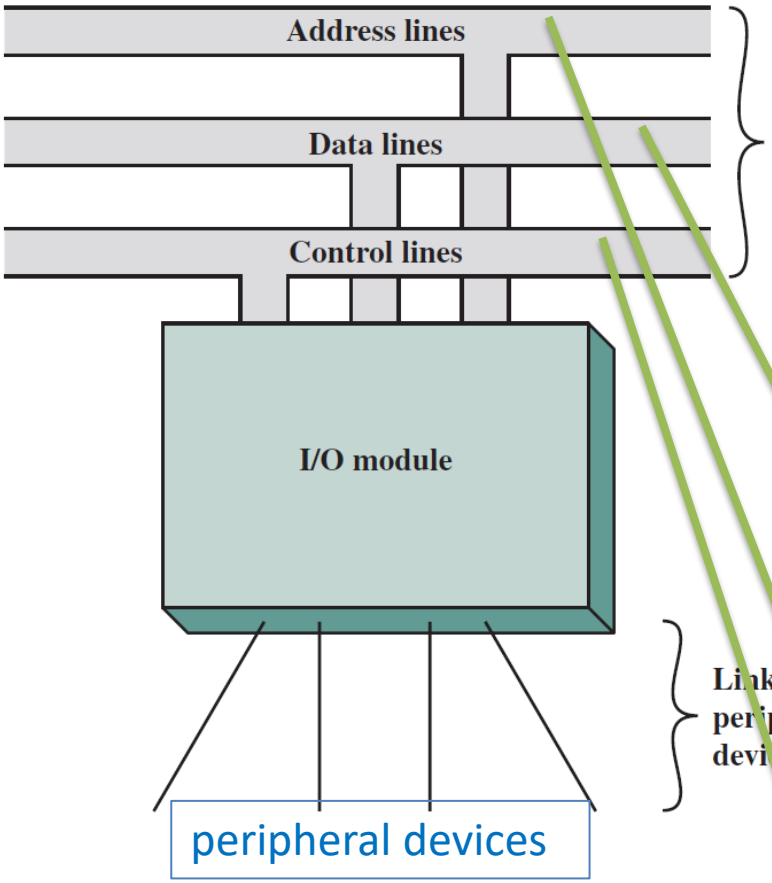
Input / Output (I/O)

- Why not connect peripherals directly to system bus?
  - Wide variety of peripherals with different data formats and length.
  - Speed mismatch between peripherals and memory/CPU leads to inefficiencies.
- An I/O module is required to perform communication and interface between peripheral devices ↔ system bus ↔ CPU/memory.

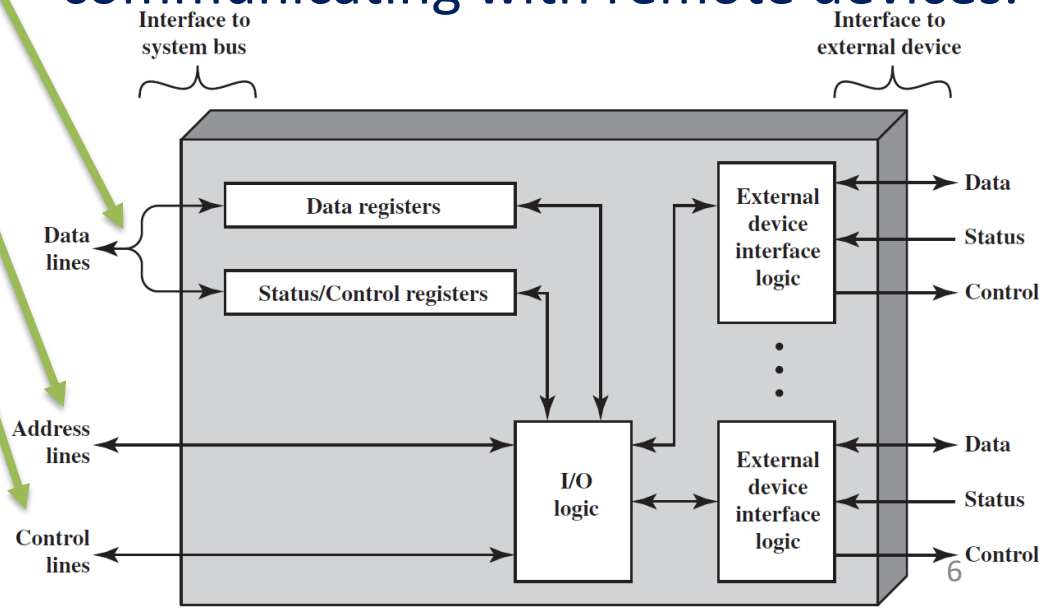
University of Glasgow

School of Computing Science

External Devices



- **Human readable**: Suitable for communicating with computer users;
- **Machine readable**: Suitable for communicating with equipment;
- **Communication**: Suitable for communicating with remote devices.



I/O modules contain logic and interface between the bus and peripherals.

Three Techniques of I/O Operations

- Programmed I/O:
  - Data are exchanged between CPU and I/O modules.
  - CPU executes a program to direct control of I/O operation.
  - CPU issues a command, and must wait until I/O operation complete.
  - If CPU is faster than I/O module, it is wasteful of CPU time.
- Interrupt-driven I/O:
  - CPU issues an I/O command to I/O module, then continues to execute other instructions.
  - It is interrupted by I/O module when the task is completed.
- Direct memory access (DMA):
  - I/O module and main memory exchange data directly without CPU involvement.

University of Glasgow

School of Computing Science

I/O Techniques to Access Memory Data

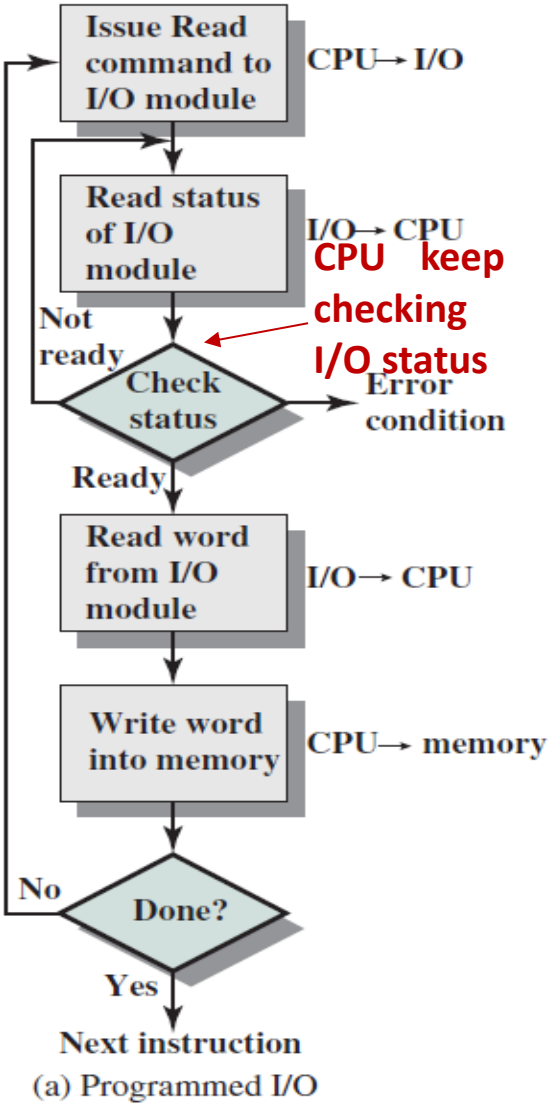
	No Interrupts	Use of Interrupts
I/O-to-memory transfer via CPU	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

- Method 1: Programmed I/O and Interrupt-driven I/O:
  - CPU is responsible to extract data from main memory and output to I/O modules.
  - CPU receives data from I/O modules and stores in memory.
- Method 2: Direct memory access (DMA):
  - CPU initializes transfer by supplying starting address and number of data to be transferred; then executes other tasks.
  - I/O module and main memory then exchange data directly, without CPU involvement. Interrupt CPU once complete.

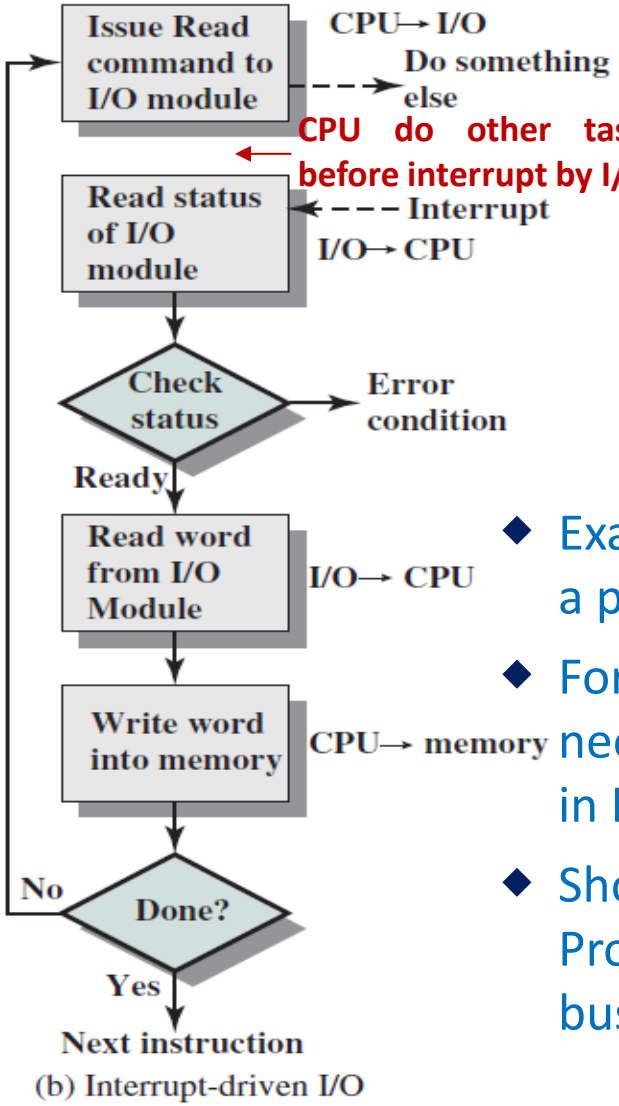
University of Glasgow

School of Computing Science

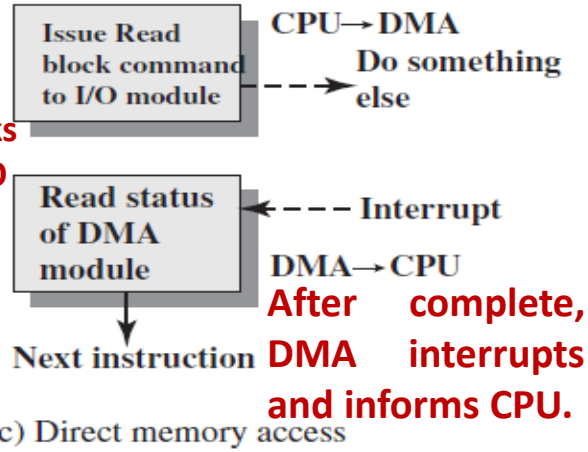
Example 4.1 - Flowchart to Read Data from I/O



(a) Programmed I/O



(b) Interrupt-driven I/O



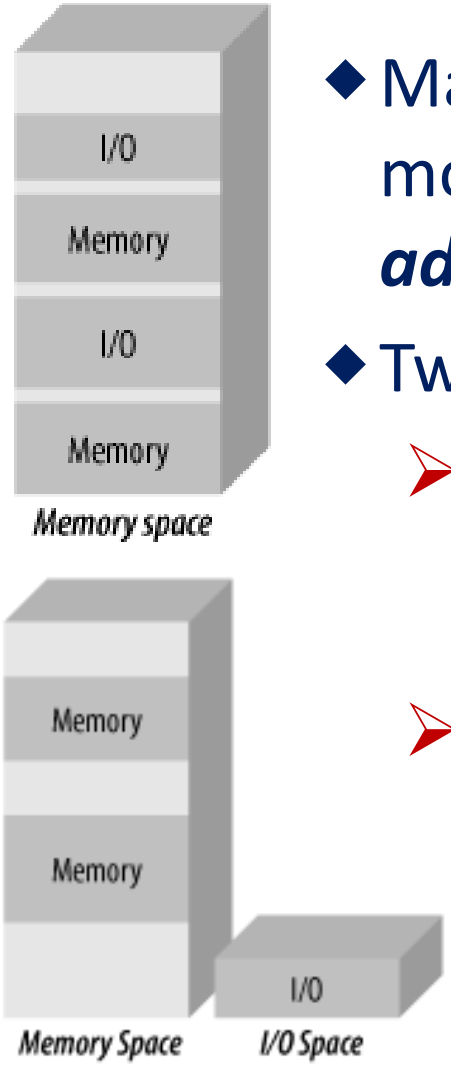
(c) Direct memory access

- Example to read in data from a peripheral into memory.
- For each word read in, CPU need check if word is available in I/O module's data register.
- Shows disadvantage of Programmed I/O, keeps CPU busy in status-checking cycle.

I/O Commands

- For I/O related instructions, CPU issues an address, specifying an I/O module, external device (peripheral), and I/O command.
- Four types of I/O commands from CPU:
  - Control: activate a peripheral and tell it what to do.
  - Test: test status conditions of an I/O module and peripherals.
  - Read: Get data from a peripheral → data registers of I/O module → data bus.
  - Write: Get data from data bus → data registers of I/O module → a peripheral.

I/O Instructions

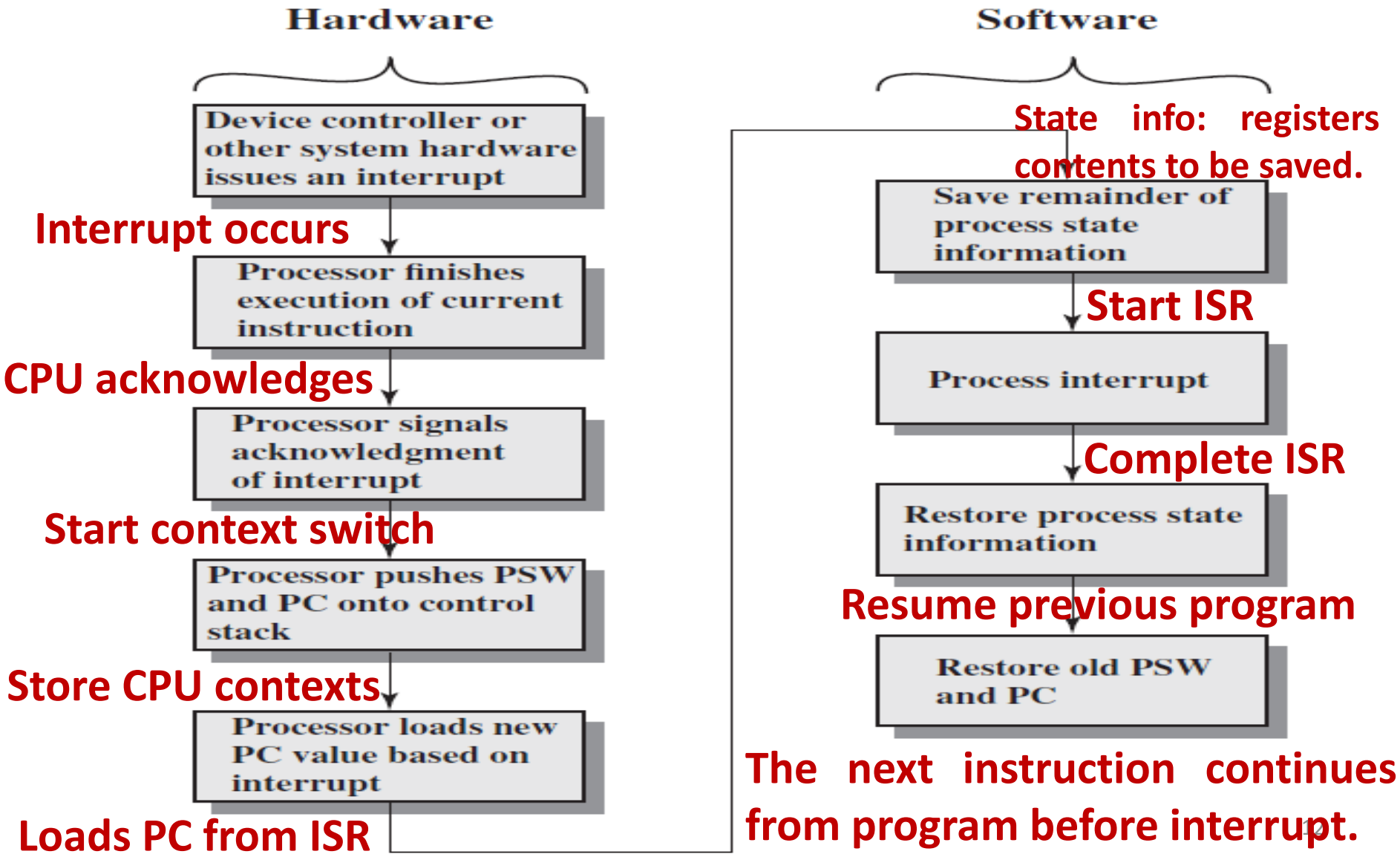


- Many I/O devices connected through I/O modules to CPU. Each device is given a unique ***address***.
- Two possible addressing modes:
  - memory-mapped I/O: a single address space for memory and I/O devices. CPU uses the same way to access both memory and I/O devices.
  - Isolated I/O: separate address spaces for I/O and memory. Need select lines to indicate if address to I/O or memory. Need special commands for I/O.

University of Glasgow

School of Computing Science

Sequence of Events in Interrupt Processing

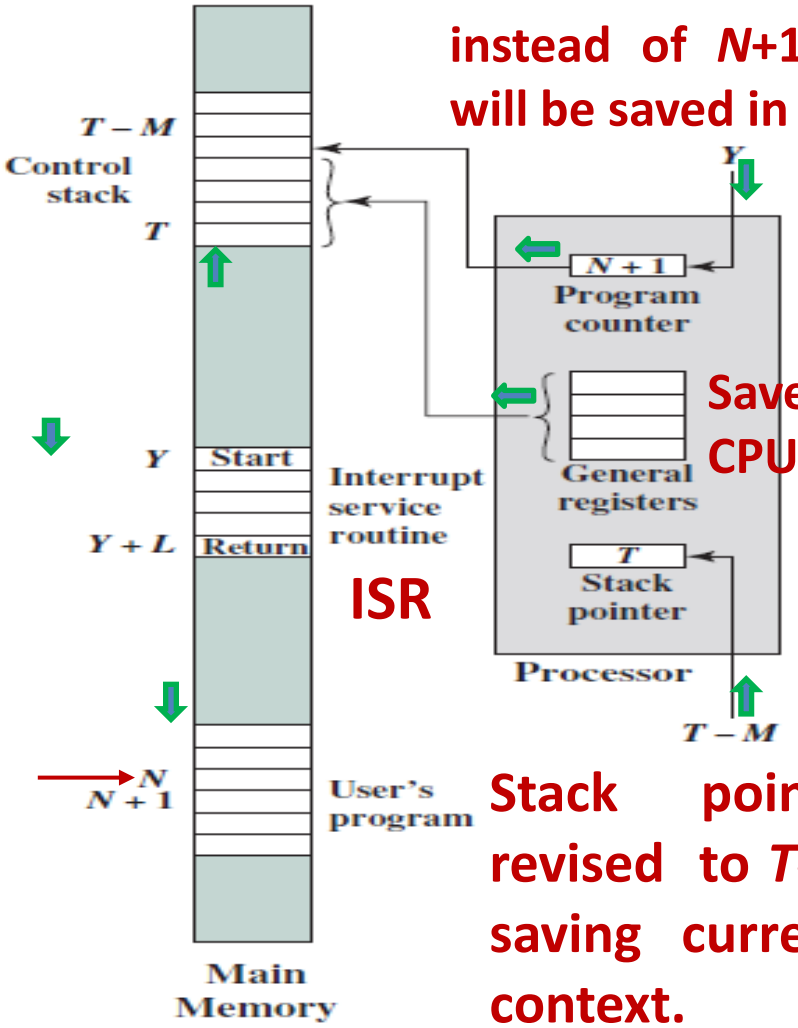


University of Glasgow

School of Computing Science

Example 4.2 – Context Switch at Interrupt

The next PC will be *Y*, instead of *N*+1. *N*+1 will be saved in stack.

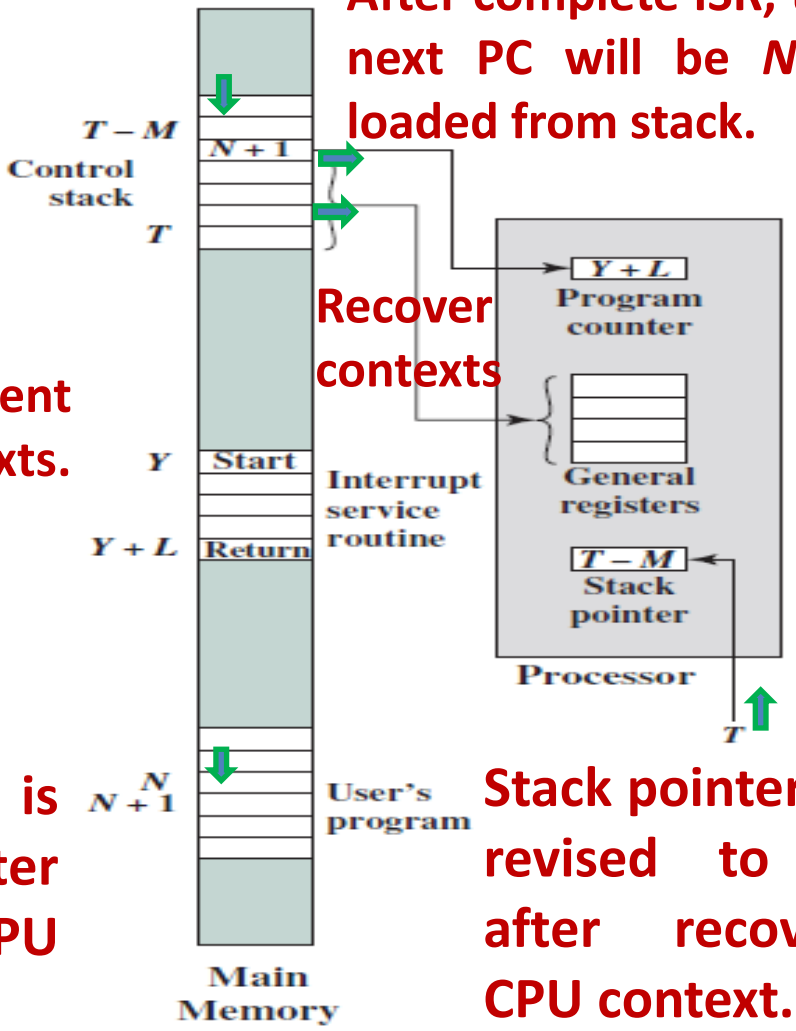


Save current CPU contexts.

Stack pointer is revised to *T-M* after saving current CPU context.

(a) Interrupt occurs after instruction at location *N*

After complete ISR, the next PC will be *N+1*, loaded from stack.



Recover contexts

Stack pointer is revised to *T* after recover CPU context.

(b) Return from interrupt

University of Glasgow

School of Computing Science

Example 4.3 – Different I/O Operations

- A system is controlled by an operator through commands entered from a keyboard. The average number of commands in 8-hour interval is 60.

  a). Suppose CPU scans keyboard every 100 ms as a programed I/O. How many times will keyboard be checked in an 8-hour period?

  b). By what fraction would the number of CPU visits to keyboard be reduced if interrupt-driven I/O were used?

Solution:

a) CPU scans keyboard every 100 ms, meaning 10 times per second. In 8 hours, the number of keyboard scans by CPU will be:

8 * 60 * 60 * 10 = 288,000.

b) If interrupt-driven I/O is used, keyboard will interrupt CPU when a new command is entered. CPU does not need scan keyboard regularly. CPU only need 60 visits to keyboard in 8 hours. Thus, the reduction of visits will be 1 – (60 / 288,000) * 100% = 99.9%.

Drawbacks of Programmed & Interrupt-Driven I/O

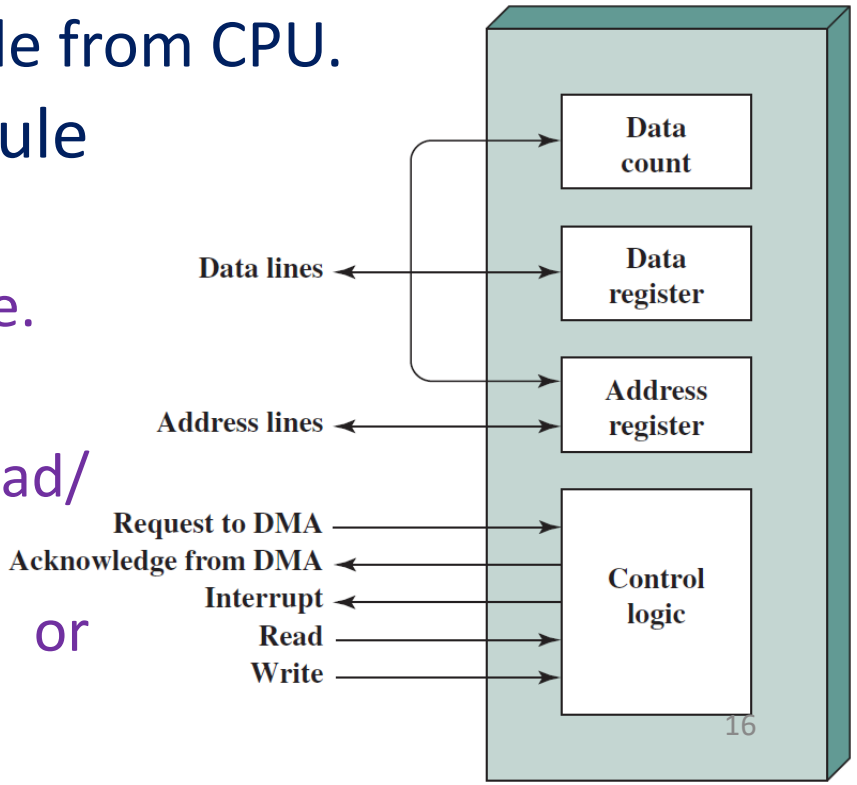
- Interrupt-driven I/O, though more efficient than programmed I/O, still requires active intervention of CPU.
- Any data transfer between memory and I/O module must traverse a path through CPU.
- CPU is tied up in managing an I/O transfer, transfer rate is limited.
- To move large volumes of data, a more efficient technique is ***direct memory access*** (DMA).

University of Glasgow

School of Computing Science

DMA Functions and DMA Block Diagram

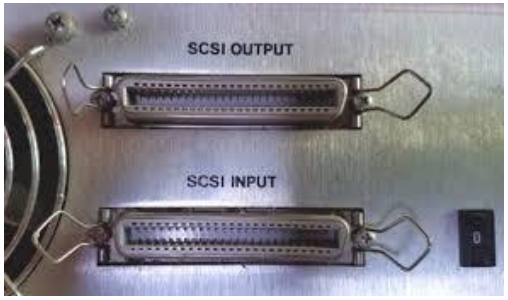
- ***Burst Mode***: DMA module takes over the bus from CPU, to transfer *a block of data*, then returns control to CPU.
- ***Cycle Stealing Mode***: DMA module takes over the bus for *each byte of data* transfer, then return control to CPU. DMA module in effect steals a bus cycle from CPU.
- CPU sends to DMA module following info:
  - Asserts read or write control line.
  - Address of I/O device involved.
  - Starting memory address to read/ write.
  - Number of words to be read or written.



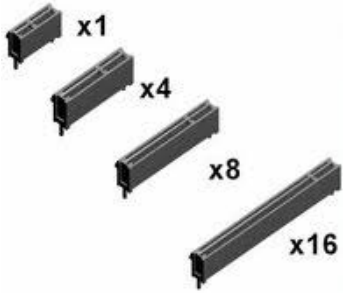
University of Glasgow

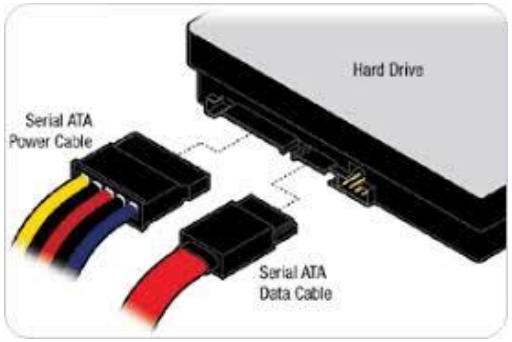
School of Computing Science

Some External Interconnection Standards



- Universal Serial Bus (USB): USB 1.0, 2.0. 3.0.
- IEEE standard 1394 or FireWire.
- Small Computer System Interface (SCSI).
- Thunderbolt.
- InfiniBand.
- PCI Express (PCIe).
- Serial Advanced Technology Attachment (SATA).
- Ethernet.
- WiFi.



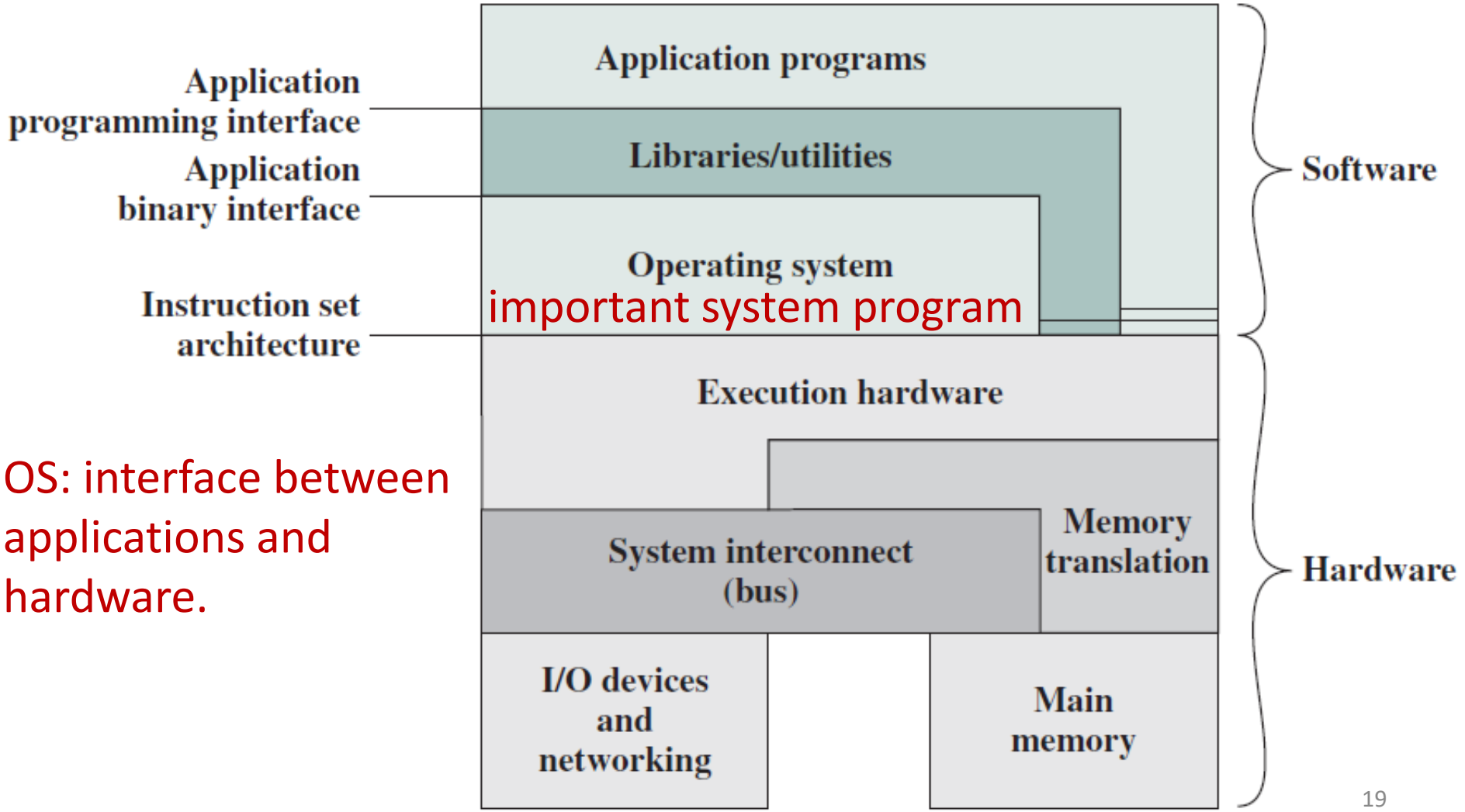


Operating System (OS)

University of Glasgow

School of Computing Science

Operating System (OS) Overview



OS: interface between applications and hardware.

Services Provided by Operating System (OS)

- OS manages computer's resources, provides services, and schedules programs execution.
  - Process Management
  - Memory Management
  - Controlled access to I/O devices
  - Controlled access to files
- Error detection and response
- Accounting

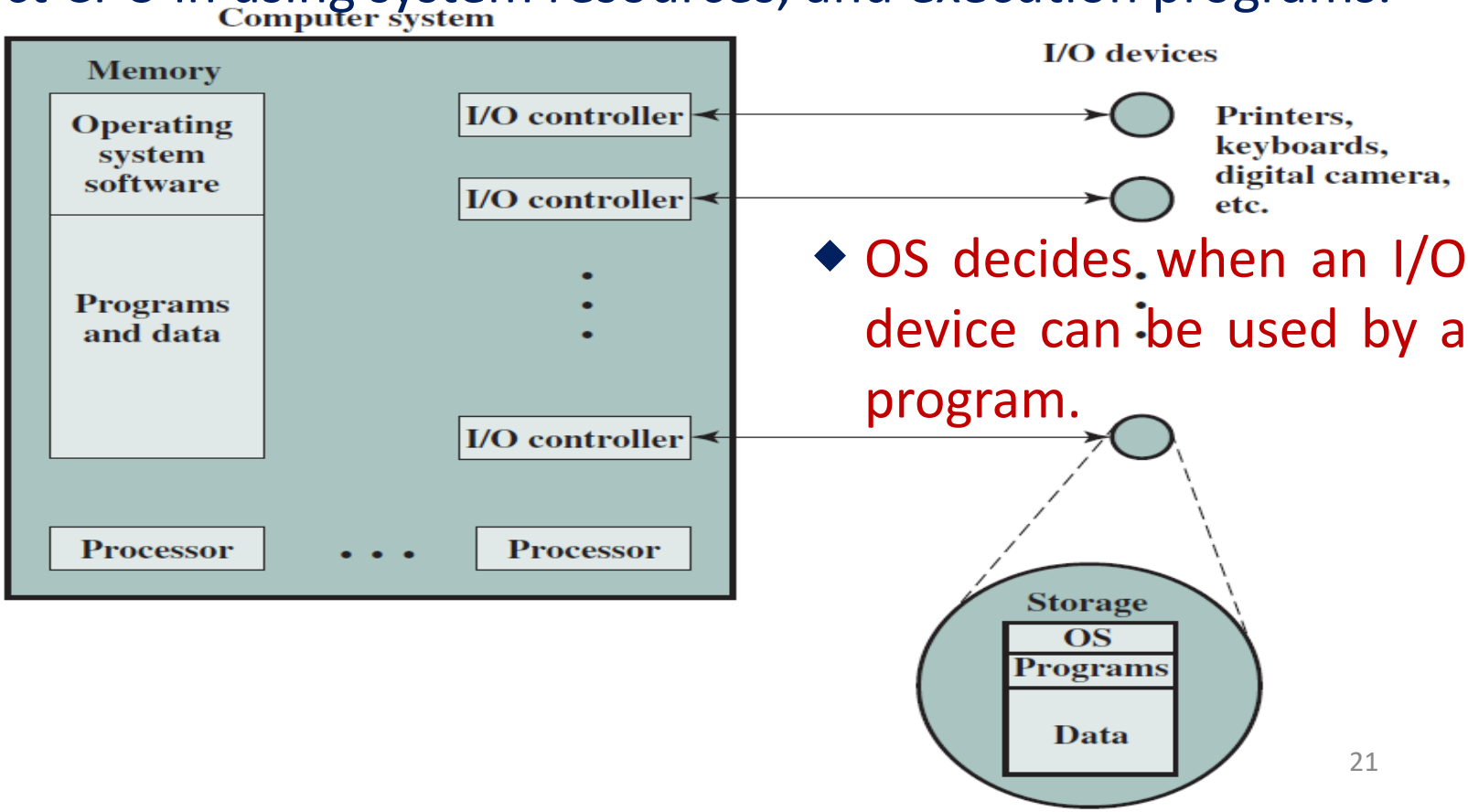
University of Glasgow

School of Computing Science

OS as Resource Manager

- Manage resources for movement, storage, and processing of data, etc.
- Direct CPU in using system resources, and execution programs.

Kernel, a portion of OS is in memory



- OS decides when an I/O device can be used by a program.

Process and Scheduling

- ***Process***: or *job*. A program in execution by CPU.
- The key to multiprogramming is *scheduling*:

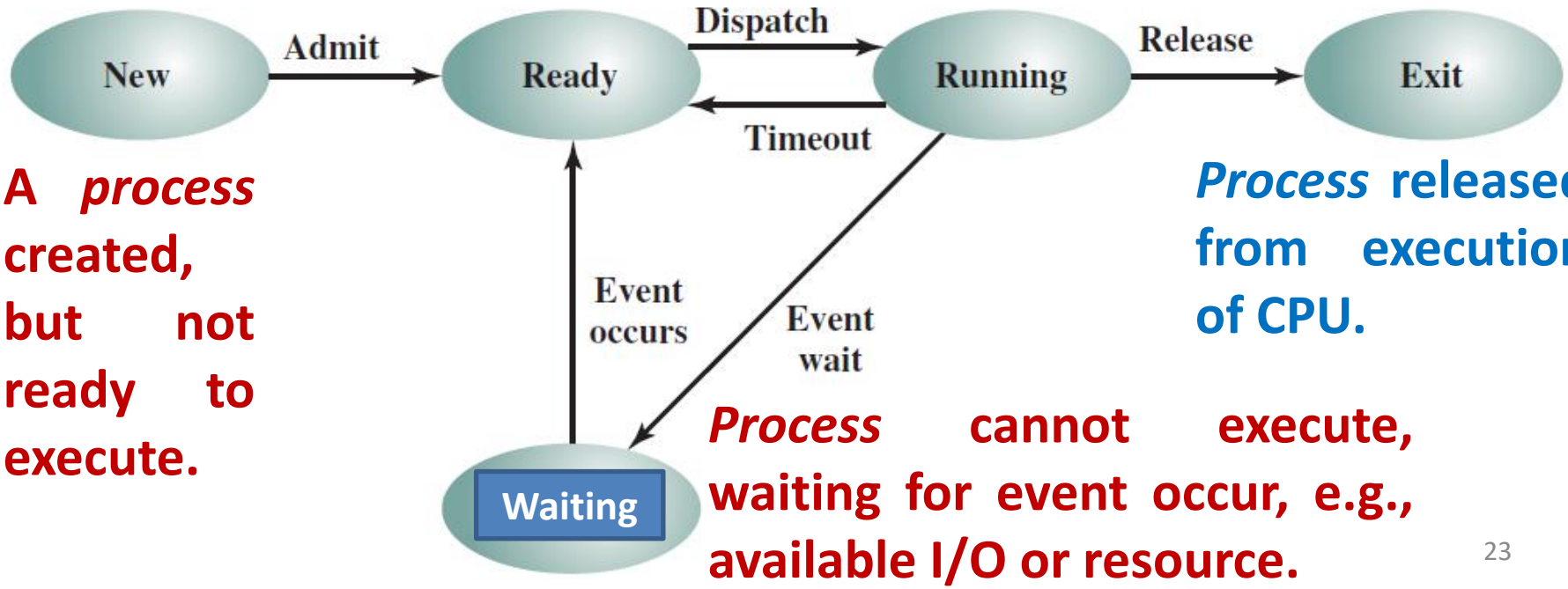
Long-term scheduling	• performed when a new process is created; • decide if there's enough memory, allow new programs added.
Short-term scheduling (Dispatcher)	• selects from processes that are ready to execute. • executes most frequently to select which job to execute next.

University of Glasgow

School of Computing Science

Five-State Process Model in OS

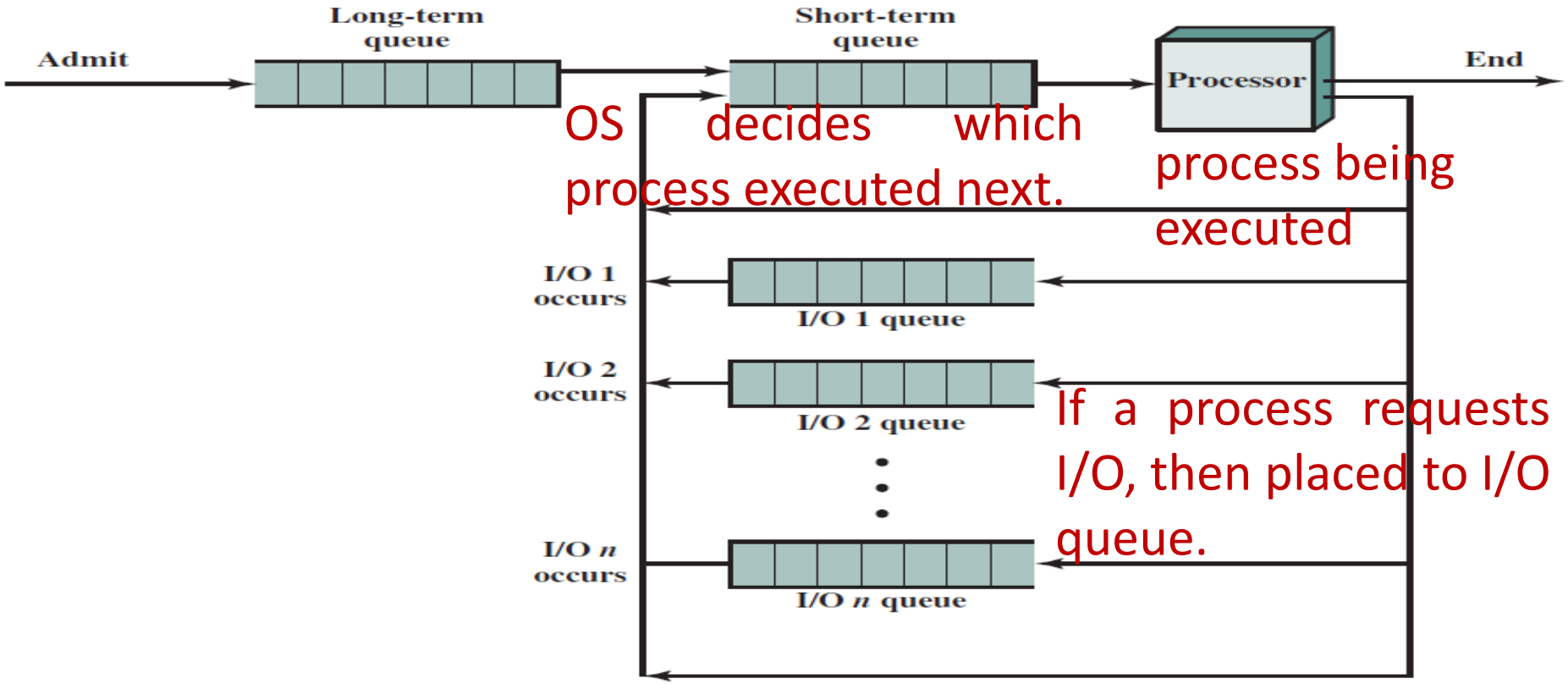
- State: during lifetime of a process, its status changes a number of times. Five defined *states* for a process:



University of Glasgow

School of Computing Science

Processes Progress in OS



OS decides which process executed next.

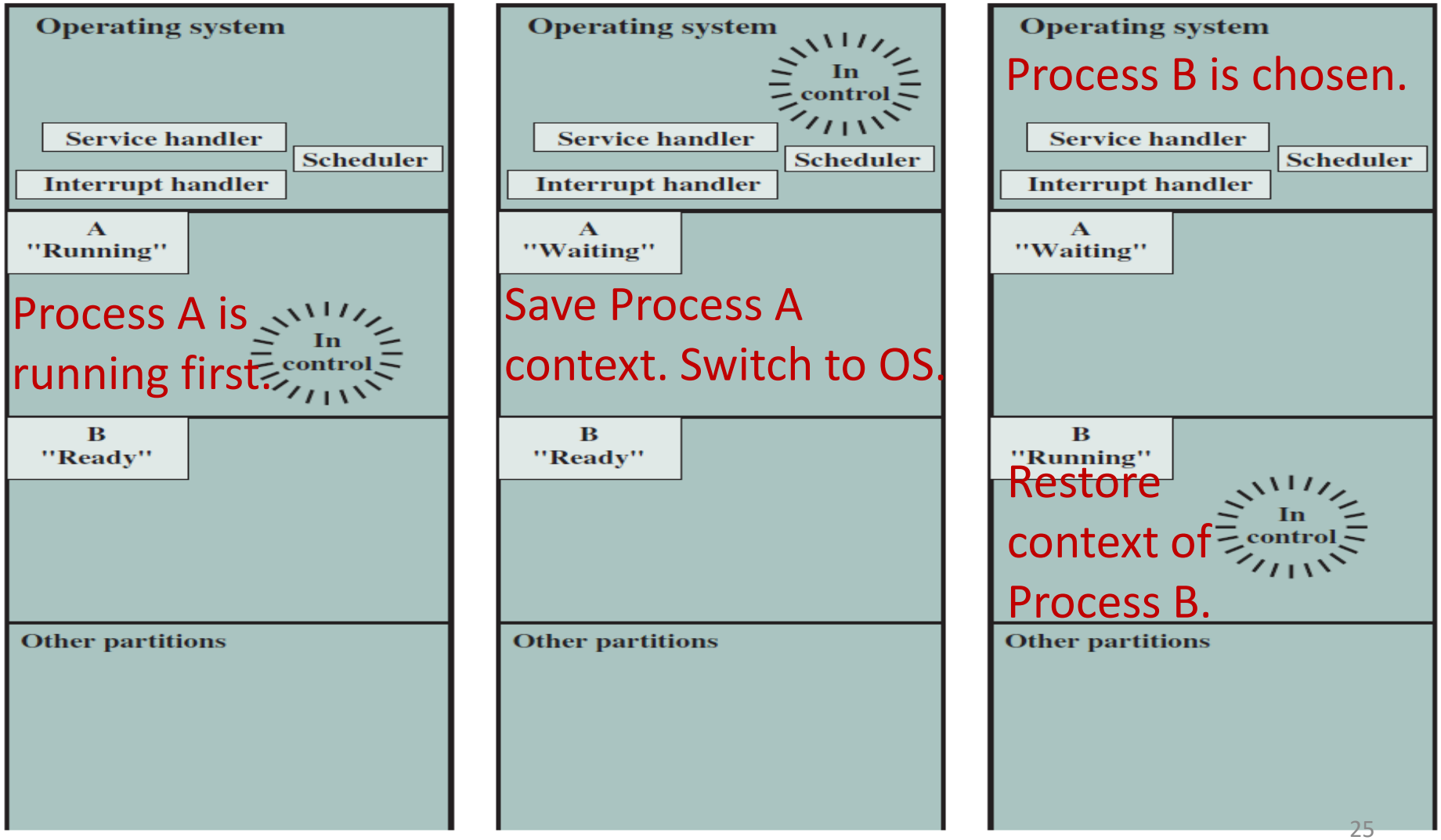
process being executed

If a process requests I/O, then placed to I/O queue.

University of Glasgow

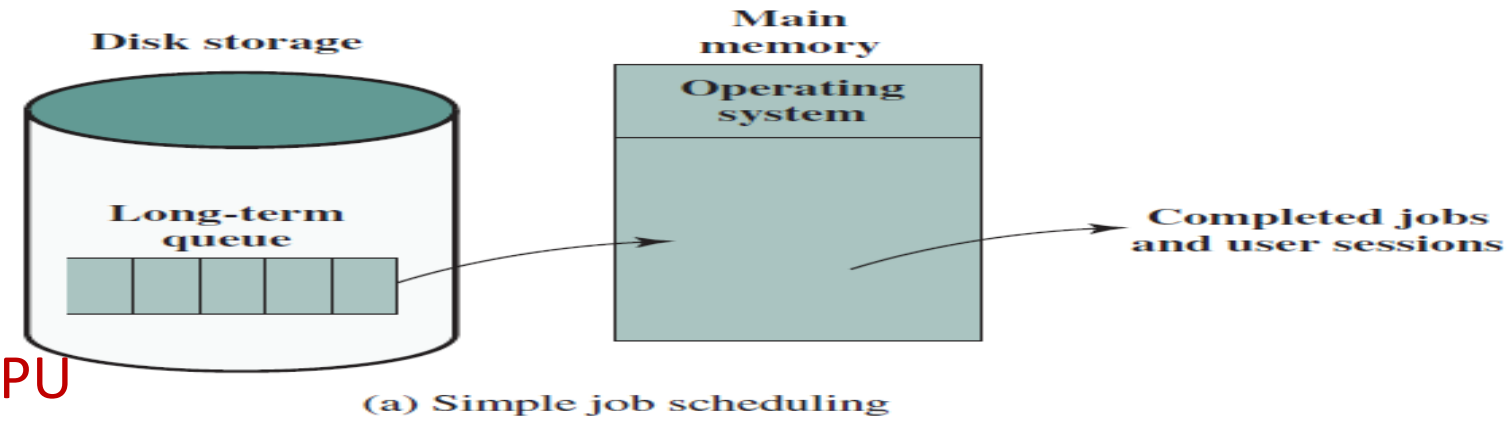
School of Computing Science

Example - Scheduling in OS



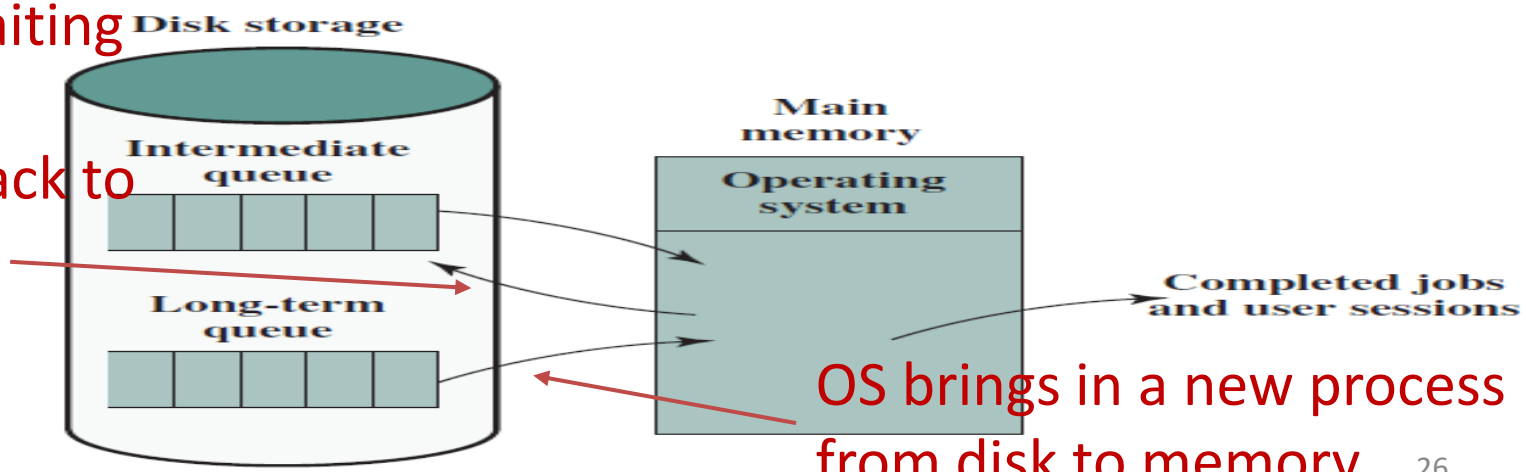
Memory Management and Swapping

- **Memory Management**: memory is divided for multiple processes, carried out dynamically by OS.



(a) Simple job scheduling

Swapping: CPU *swaps* a waiting process in memory back to disk.



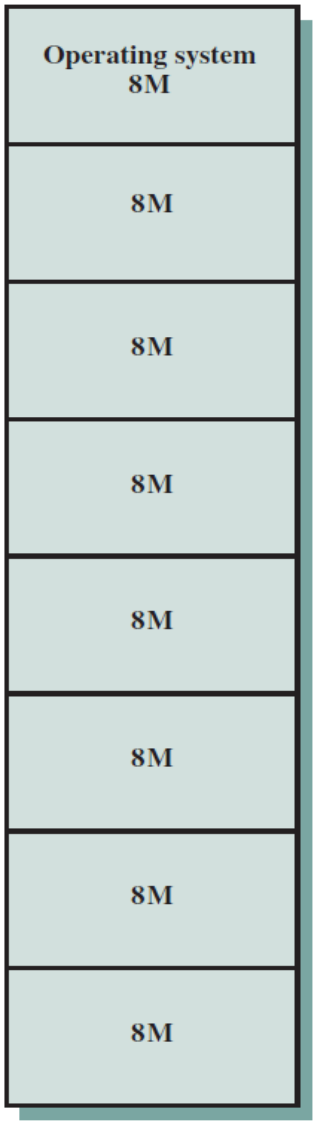
(b) Swapping

OS brings in a new process from disk to memory.

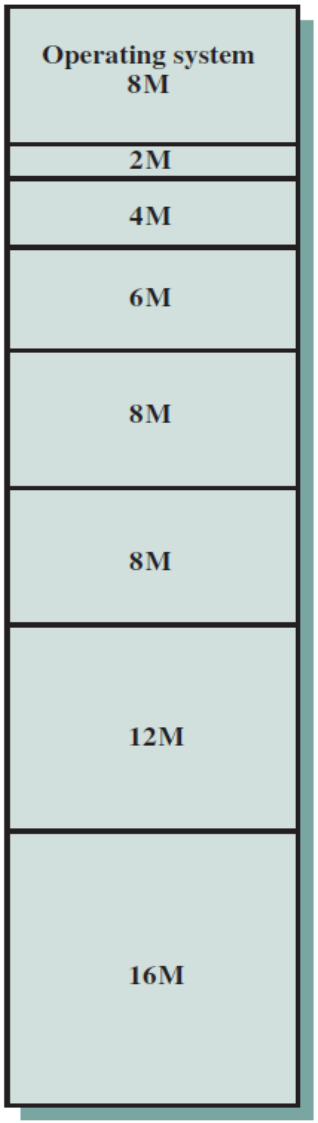
University of Glasgow

School of Computing Science

Partitioning of Memory



(a) Equal-size partitions



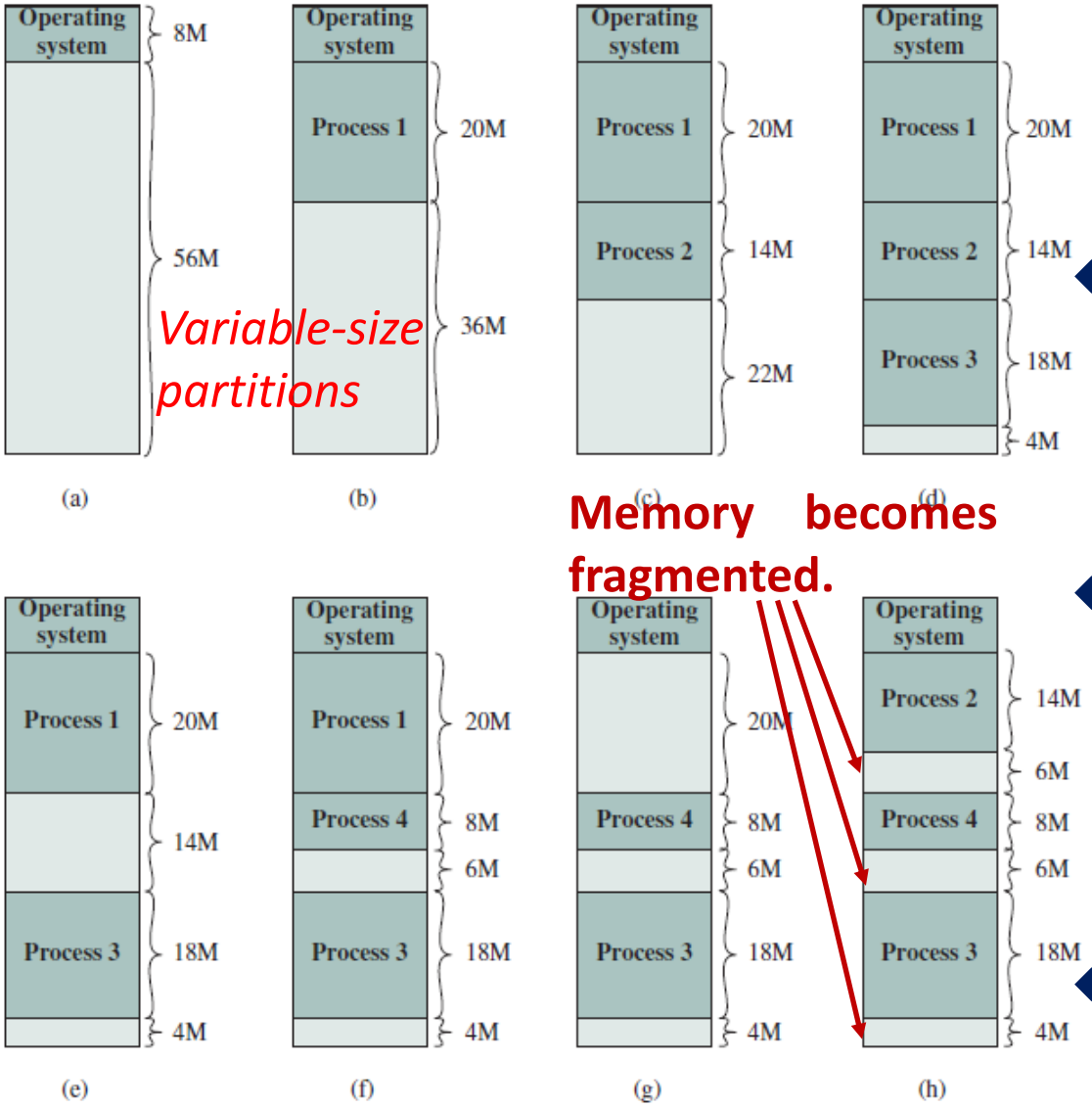
(b) Unequal-size partitions

- *Fixed-size partitions:* Fixed size need not be of equal size.
- Unequal fixed-size partitions still with wasted memory. A process not use exactly provided memory.

University of Glasgow

School of Computing Science

Dynamic Partitioning



Variable-size partitions

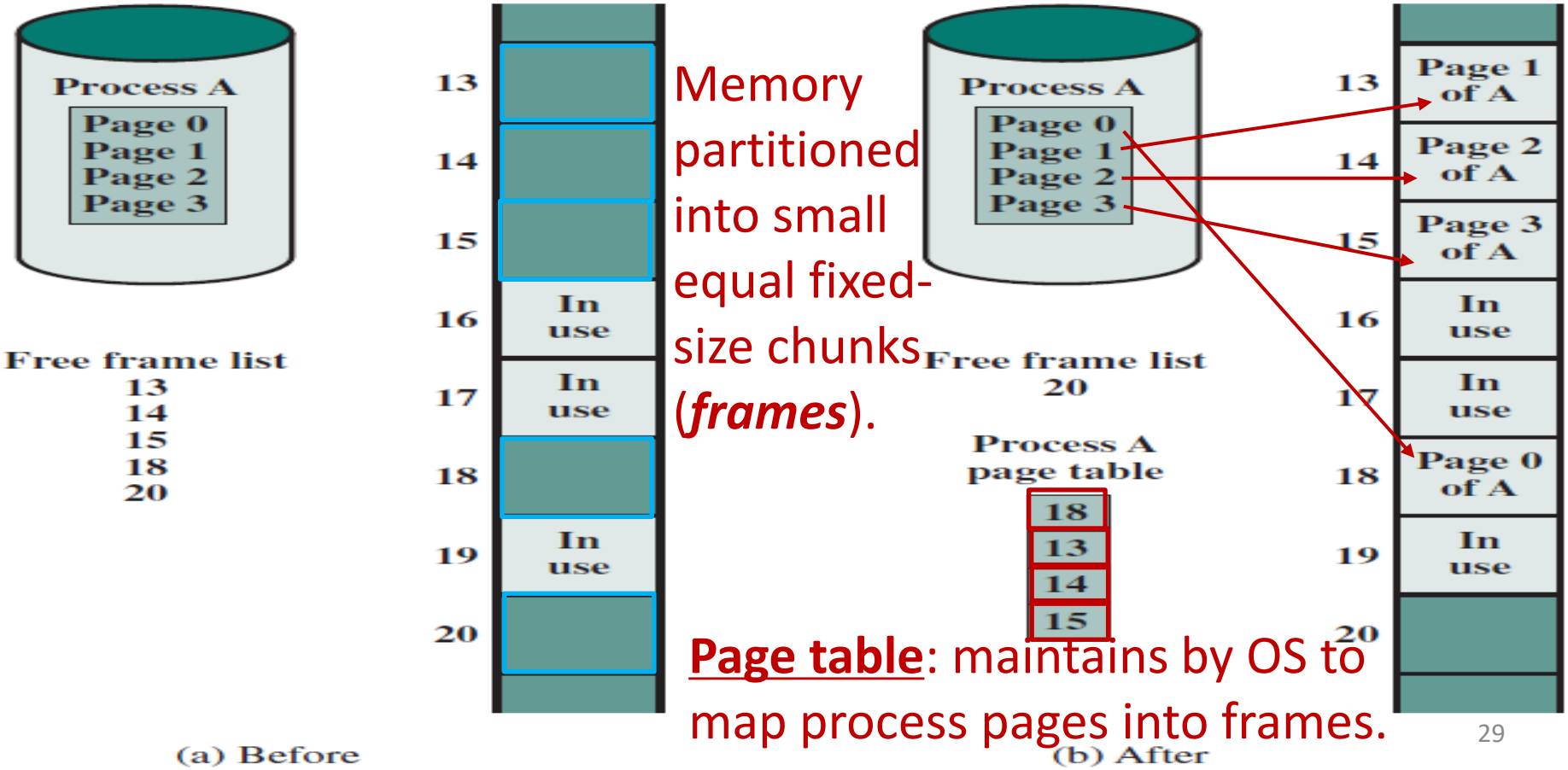
Memory becomes fragmented.

- Variable-size partitions: Leads to many small holes in memory (fragmented); memory utilization ⬇.
- ***Compaction***: OS shifts processes in memory to consolidate all free memory in blocks.
- ***Logical address***: relative to the beginning of a program. Instructions in program contain only logical addresses.
- ***Physical address***: an actual location in main memory.

Paging of Memory

Each process divided into small fixed-size chunks (***pages***).

page size = frame size

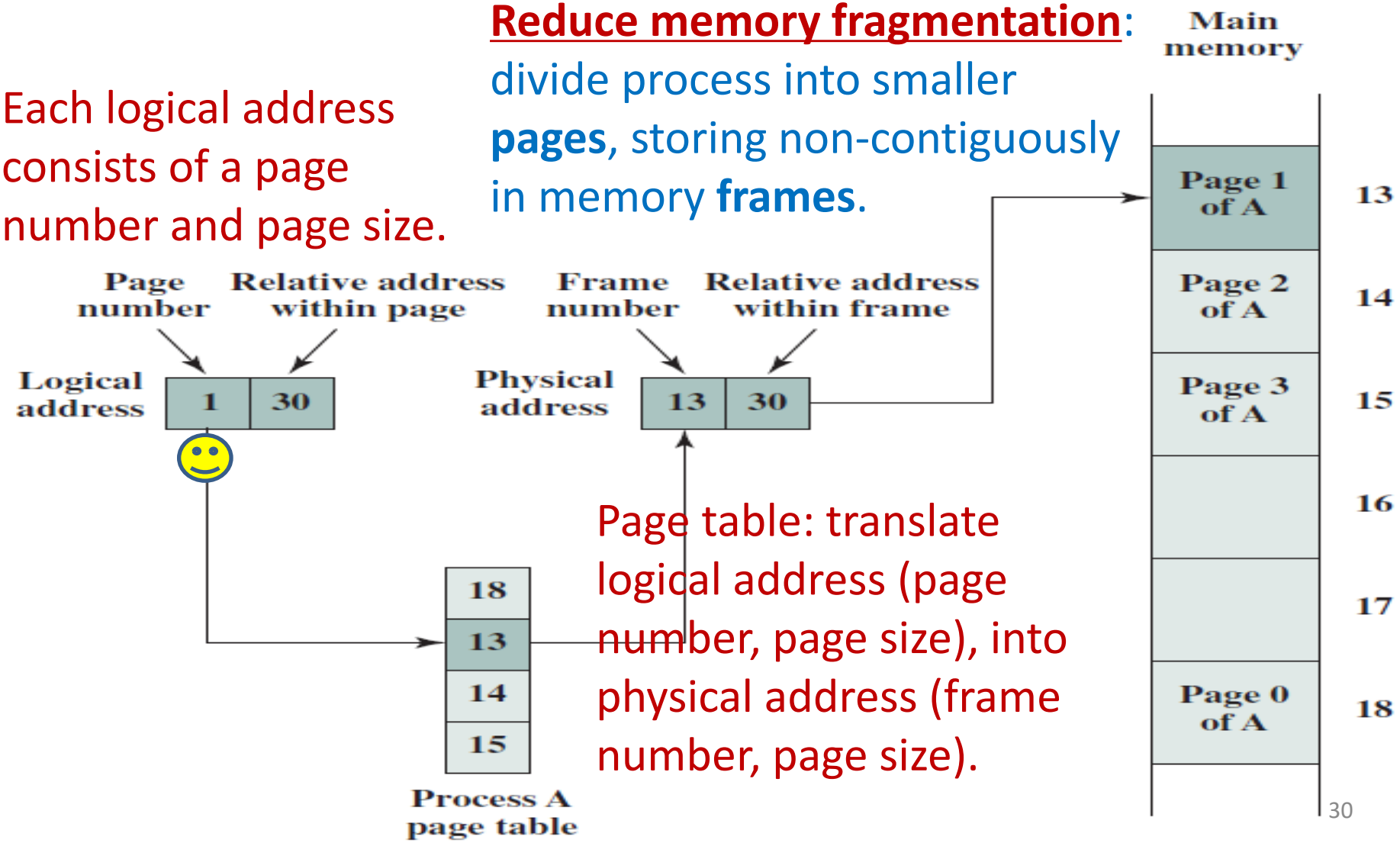


Page table: maintains by OS to map process pages into frames.

Page Table

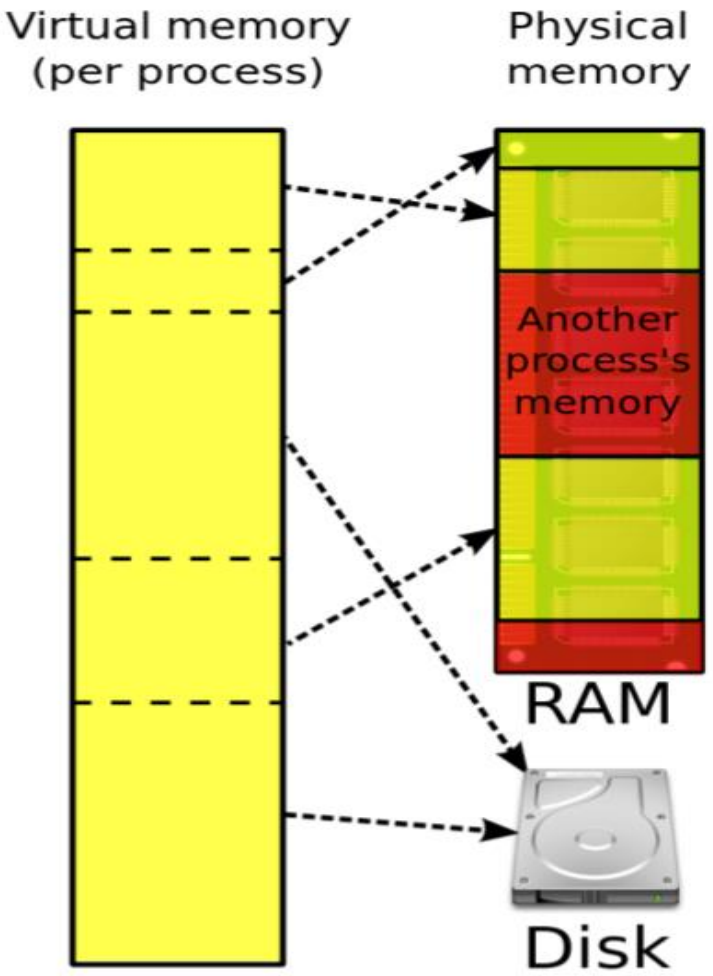
Reduce memory fragmentation: divide process into smaller **pages**, storing non-contiguously in memory **frames**.

Each logical address consists of a page number and page size.



Virtual Memory

- In reality, most processes do not need all pages at once.



- Load portions of processes in only when needed.
- Programs can use larger virtual memory space than physical space.
- Each process only uses a small space of their total address space, more memory left to store other programs.

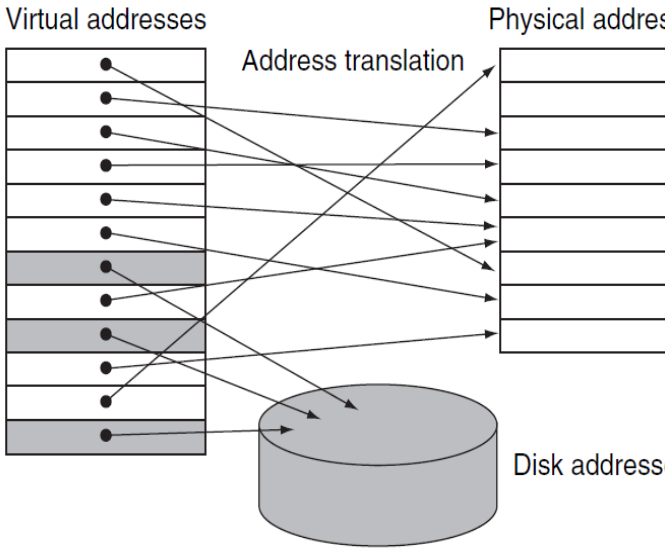
University of Glasgow

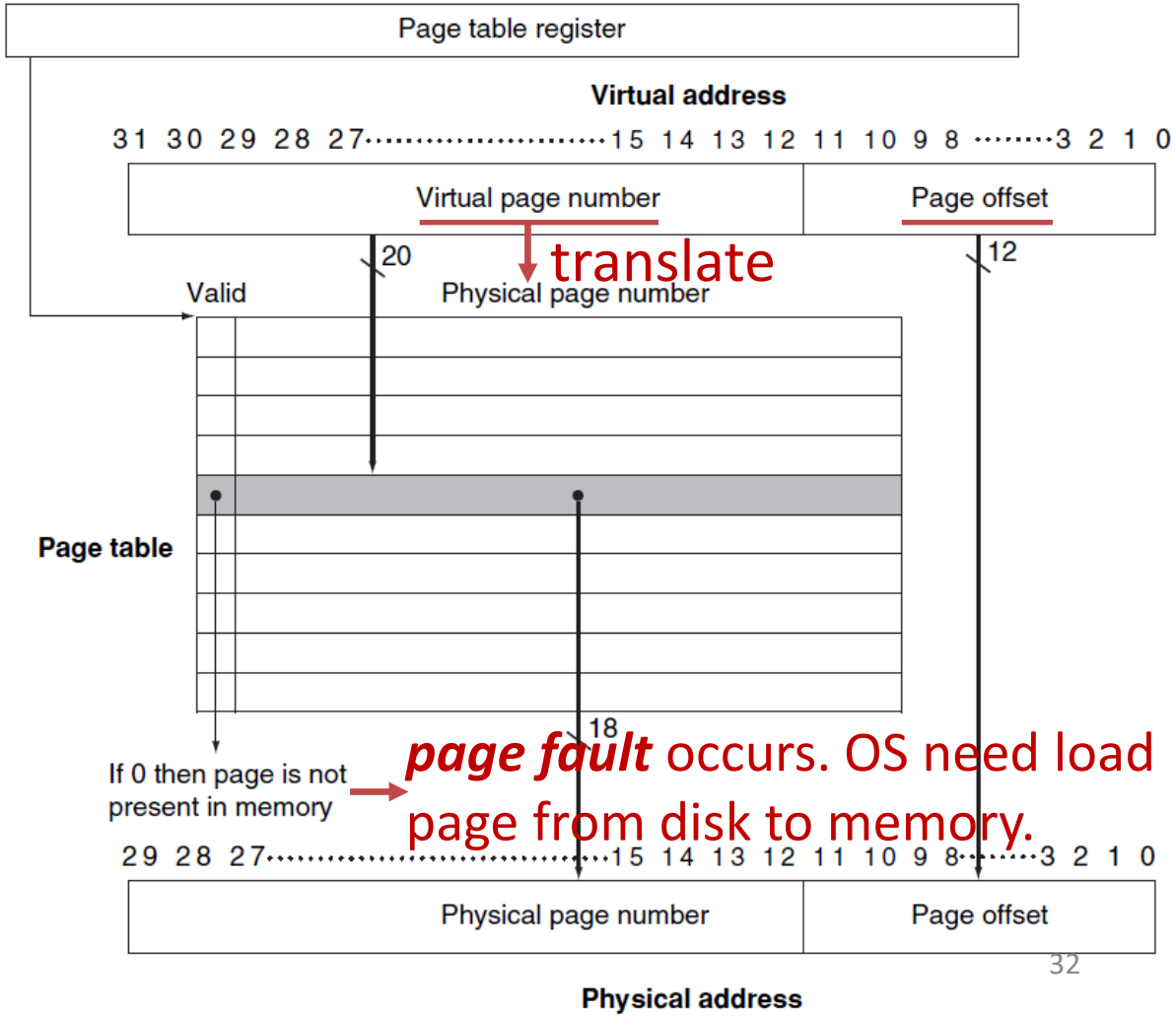
School of Computing Science

Virtual Memory

Address mapping: maps virtual page number to physical page number.

valid bit: indicates if a page is stored in main memory or not.





University of Glasgow

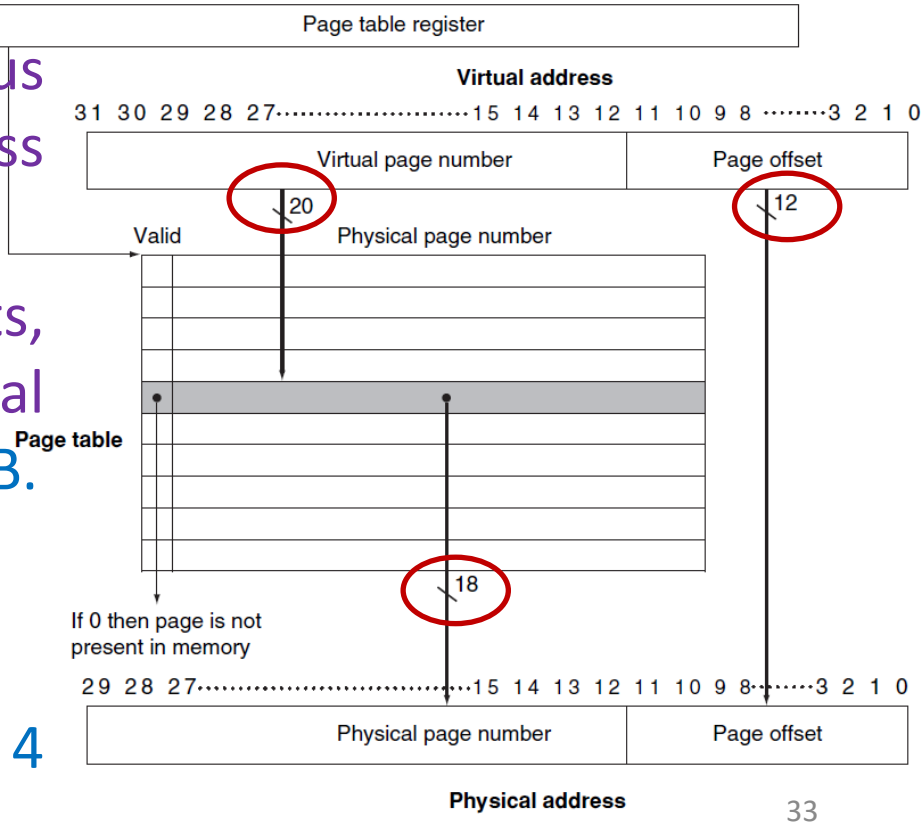
School of Computing Science

Example 4.4 – Virtual Address

- Page size (page offset) is 2^{12} bytes, or 4 KiB. Calculate what is virtual address space, and physical address space? There are 4 bytes per page table entry, compute maximum page table size?

Solution:

- Virtual page number is 20 bits, plus 12 bits for page size. Virtual address space is: $2^{(20+12)}$ bytes = 4 GiB.
- Physical page number is 18 bits, plus 12 bits for page size. Physical address space: $2^{(18+12)}$ bytes = 1 GiB.
- Number of page table entries = $2^{32} / 2^{12} = 2^{20}$
- Max size of page table = 2^{20} * 4 bytes = 4 MiB.



Next Lecture

Lecture 5: Number Systems, Computer Arithmetic and Logic