Basic Organization of Computer and Memory Management

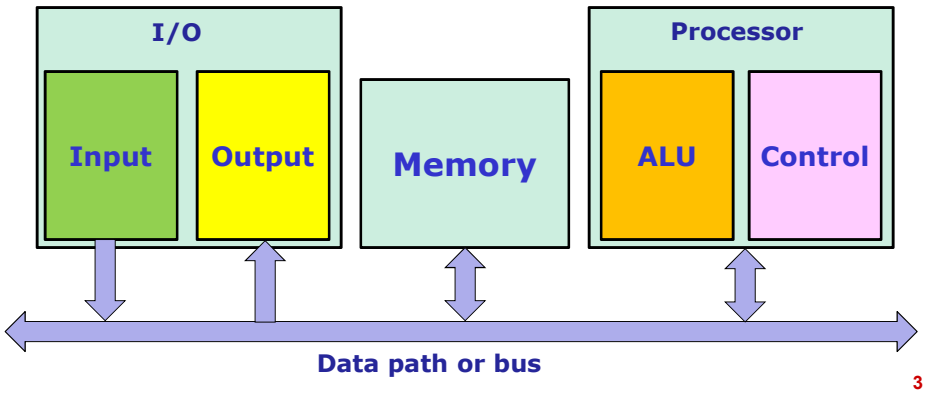
Digital Computer

- Computer is a fast electronic computing/calculating machine that
  - Accepts digitized input information
  - Processes it according to a list of internally stored instructions
  - Produces the resulting output information
- Internal storage is called computer memory
- List of instructions to perform a task is called a computer program
- Many types of computers exists that differ widely in size, cost, computational power and purpose of use

2

Microarchitecture Level

- **Functional Units**:
  - Input Unit
  - Memory Unit
  - Arithmetic and Logic Unit (ALU)
  - Output Unit
  - Control Unit



3

Information Processed by Computer

- **Instructions:**
  - Instructions are commands that
    - Govern the transfer of information with in computer as well as between the computer and its I/O devices
    - Specify the arithmetic and logic operations to be performed
  - A set of instructions that perform a task is called program
  - Usually a program is stored in memory
  - Processor fetches the instructions that make up the program from memory, one after another, perform desired operation
- **Data:**
  - They are numbers or encoded characters that are used as operands by the instructions
- Information handled by a computer is encoded in a suitable format (string of binary digits called bits – 0/1)

4

Basic Operational Concepts

- High-level program segment

```
{
    int a,b,c;
    scanf("%d, %d", a,b);
    c=a+b;
    printf("&d", c);
}
```

- Assembly level program segment

IN PORTA, LOCA - Read operand from input port and store into a memory location, LOCA

IN PORTA, LOCB

LOAD LOCA, R0 - Load the content from LOCA to processor register R0

ADDM LOCB, R0 - Add the operand at memory location LOCB to the operand in processor register R0 and places the result in R0

STORE R0, LOCC - Store the result in R0 to memory location LOCC

OUT PORTB

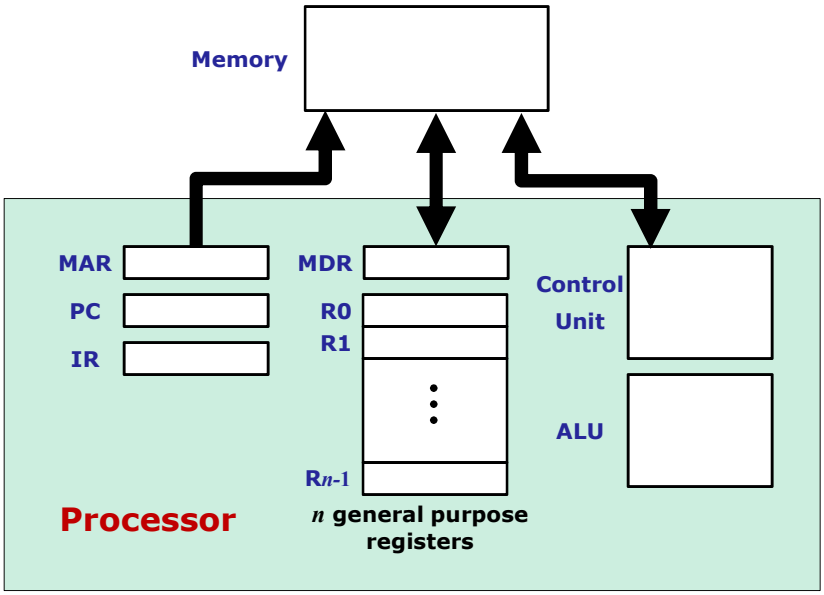
5

Execution of an Instruction

- Execution of an instruction requires to perform several steps
  - Instruction is fetched from memory into processor
  - If the instruction include operands, then the operands are fetched
  - If an instruction is for arithmetic operation, perform that operation on the fetched operands and store the results in destination location
- Transfers between memory and processor are started by sending the address of the memory location to be accessed to memory unit and issuing the appropriate control signals

6

Operational Details



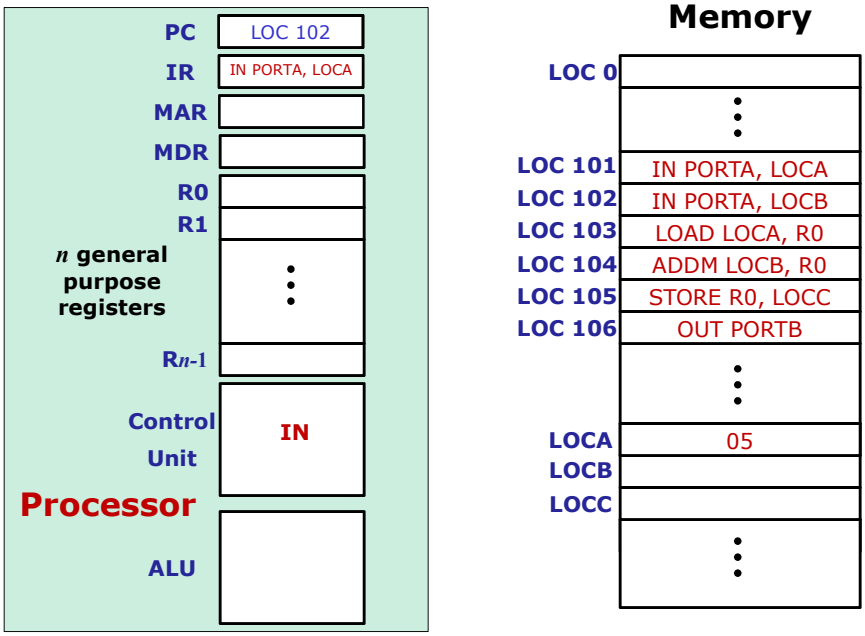
7

Registers in Processor

- General purpose registers
  - Hold the operands or address of the operand
  - Typically 16 to 32
- IR (Instruction Register)
  - Holds the instruction currently being executed
- PC (Program Counter)
  - Holds the memory address of the next instruction to be fetched and execured
- MAR (Memory Address Register)
  - Holds the address of the memory location to be accessed
- MDR (Memory Data Register)
  - Holds the data to be written into or read out of the addressed location

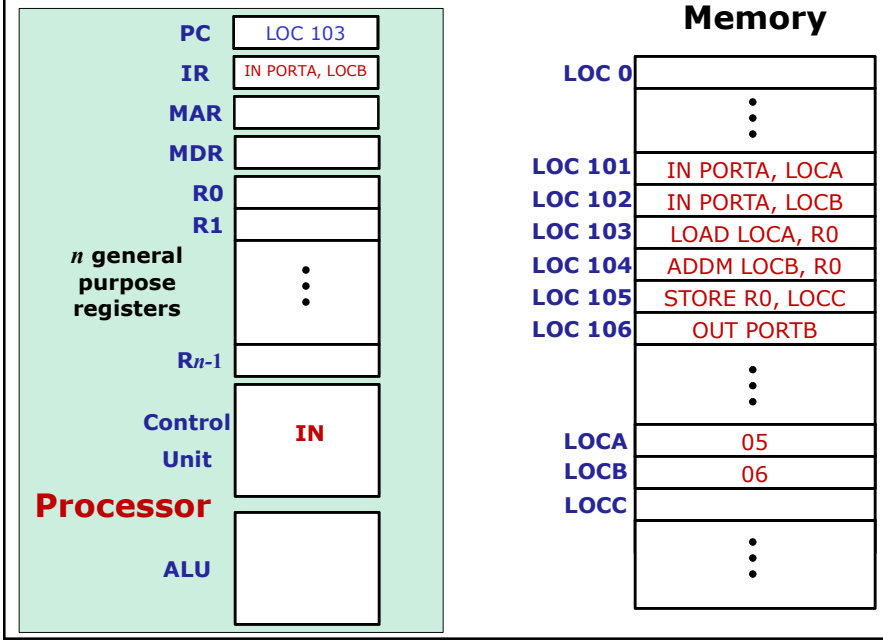
8

Execution of an Instruction



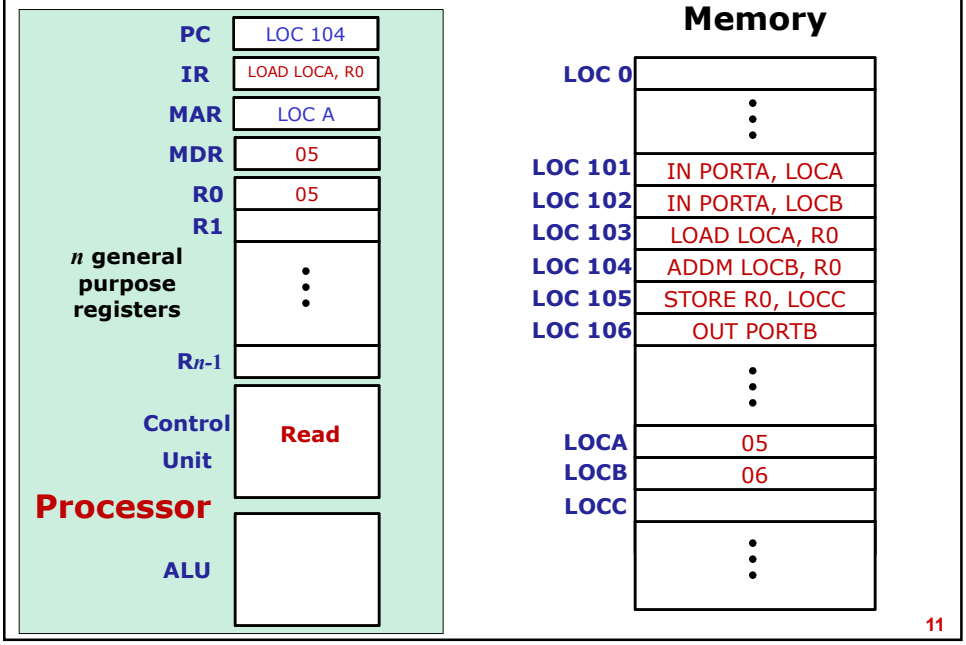
9

Execution of an Instruction

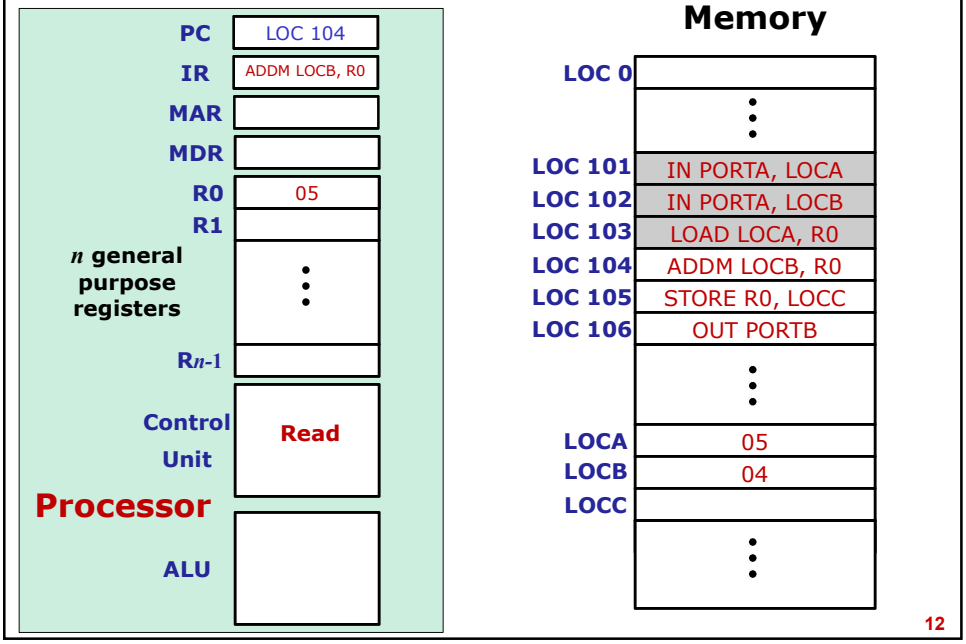


10

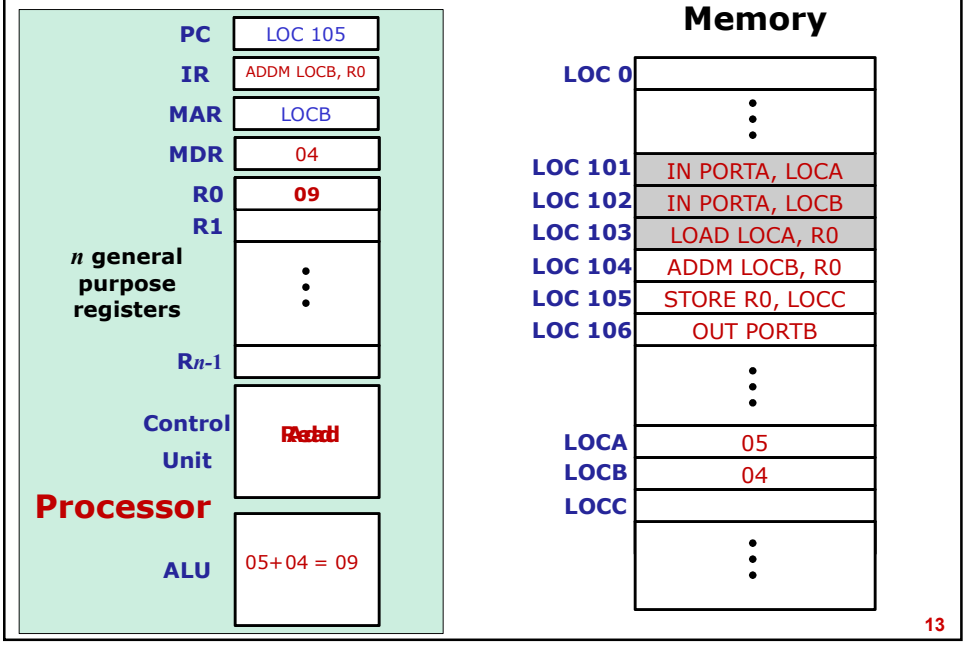
Execution of an Instruction



Fetching of an Instruction, ADDM

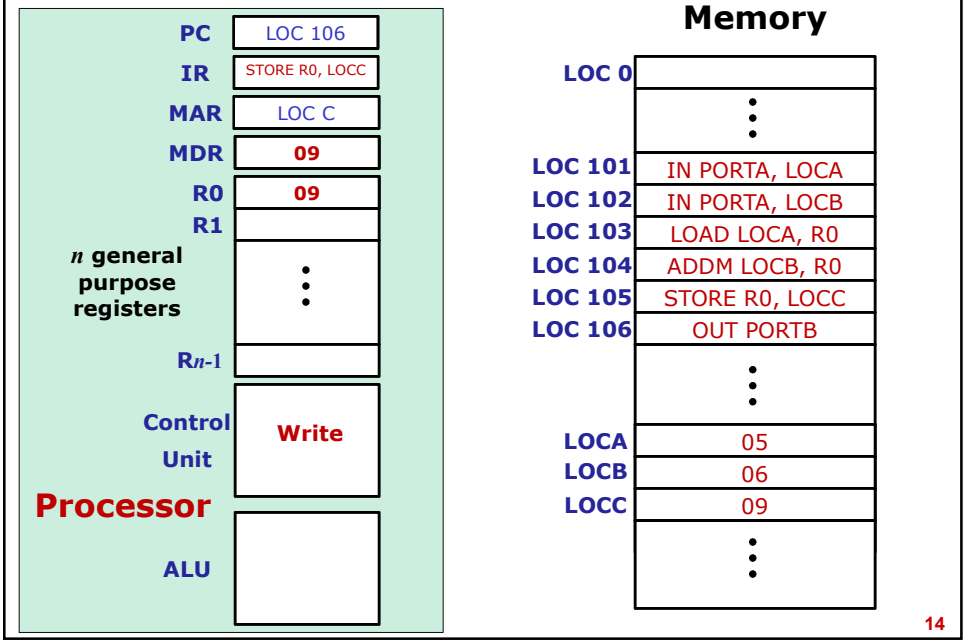


Execution of an Instruction, ADDM



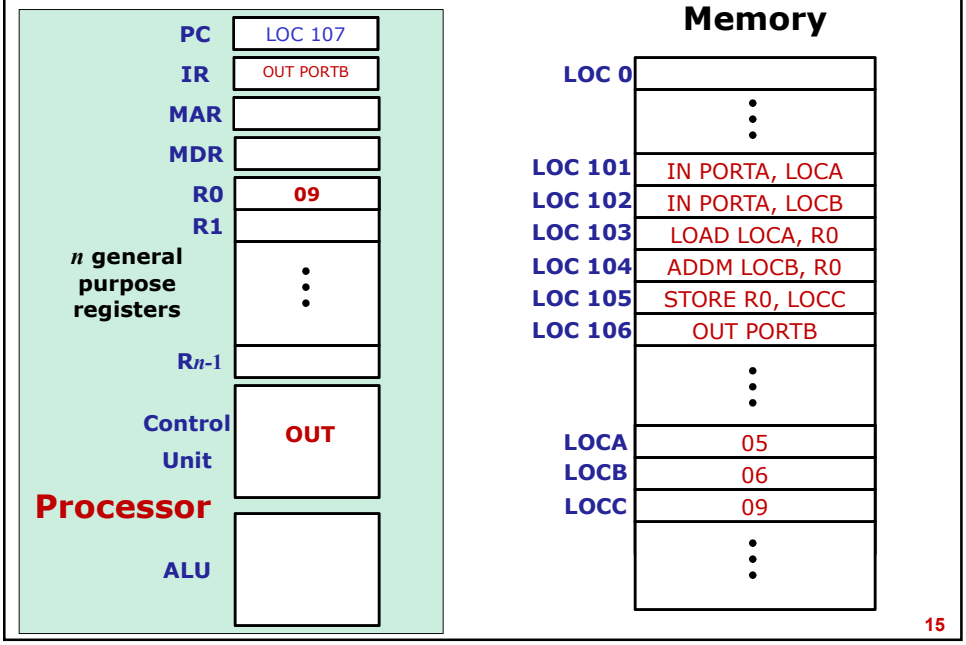
13

Execution of an Instruction



14

Execution of an Instruction



15

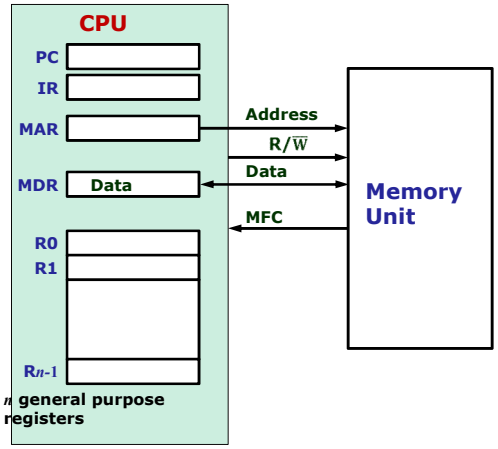
Computer Memory

Computer Memory

- Number & character operands, as well as instructions are stored in the memory of the computer
- Stored program concept
- CPU executes the instructions for which the instructions and operands have to come from memory unit
- Operations which involve memory:
  - Instruction fetch
    - Memory read
  - Memory operand fetch and store
    - Memory read
    - Memory write
- Instructions involving memory access:
  - LOAD and STORE instructions

17

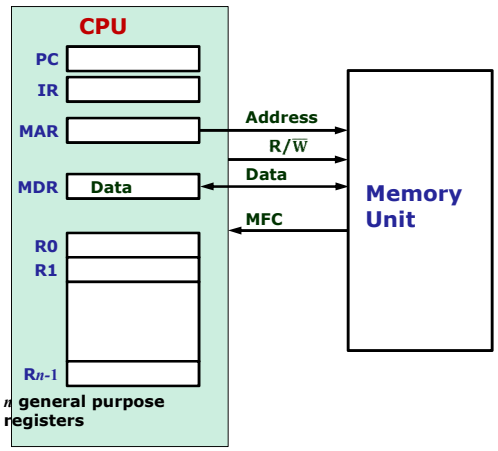
Memory Read and Write Operation



- **Read**
  - Processor loads the address of memory location into MAR
  - Set the R/$\overline{W}$ line to 1
  - Memory responds by placing the data from address location onto data line
  - Confirm the action by asserting MFC (memory function complete) signal
  - Upon receiving MFC signal, processor loads the data on data line into MDR

18

Memory Read and Write Operation



- **Write**
  - Processor loads the address of memory location into MAR
  - Processor loads data into MDR
  - Set the R/$\overline{W}$ line to 0 to indicate write operation
  - Processor places the data in MDR onto data line
  - Data on data line is written into memory location
  - Memory confirms the action by asserting MFC signal

19

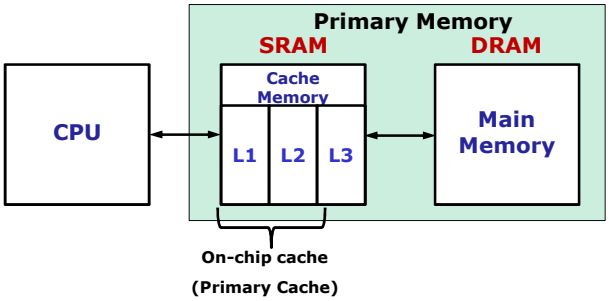
Memory Latency and Memory Organization

- **Latency**: Time to access the first of the sequence of memory words
- What is involved in determining the latency of the memory operation?
  - Processor issues the logical address to memory unit
  - The logical address need to be converted into physical address
- Memory unit is called random access memory (RAM)
  - Any location can be accessed for read/write operation independent of the location's address
- Memory unit is organised in hierarchical manner

20

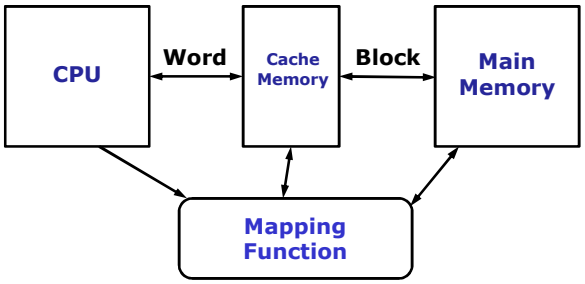
Computer Memories

- The speed of main memory is very slow in comparison with the speed of modern processors
- For good performance, processor cannot spend much of its time in waiting to access instructions and data in the main memory
- Scheme is needed to reduce the time to access information
- Efficient solution is the use of cache memory



21

Use of a Cache Memory



- Unit of transfer between main memory and cache is block
  - A block is a set of fixed number of words in contiguous address locations
  - Cache block is also called as Cache line
- Mapping function: Correspondence between main memory blocks and those in the cache

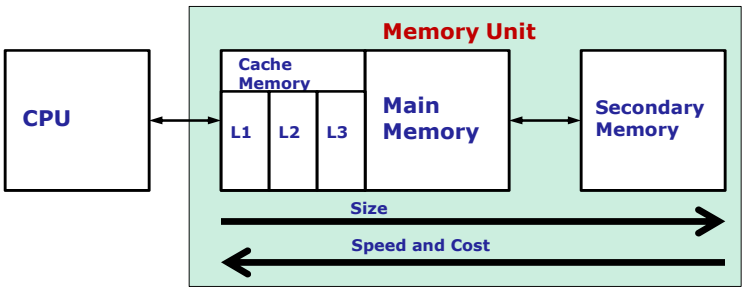
22

Cache Memory

- The cache memories are designed to exploit the locality of reference in the program
- **Locality of reference**:
  - Many instructions in localized areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently
- Different ways of locality of reference
  1. Temporal locality:
     - Recently executed instruction is likely to execute again very soon
  2. Spatial locality:
     - Instructions in close proximity to a recently executed instruction (with respect to instruction address) are likely to be executed very soon

23

Memory Hierarchy



- Processor processes instructions and data faster than it can be fetched from memory unit
- Memory access time is the bottleneck
- One way to reduce memory access time is to use faster memory
  - A small and faster memory bridge the gap between processor and main memory

24

Virtual Memory

- Ideally, entire memory hierarchy would appear to the processor as a single memory unit
- In modern computer system, the physical main memory is not as large as the address space spanned by the address issued by the processor
- When a program (or process) does not completely fits into the main memory, parts of it will be there in secondary memory
- In modern computers, operating system moves the data automatically between main memory and secondary storage
- Programmer does not need to aware of the limitations imposed by the main memory

25

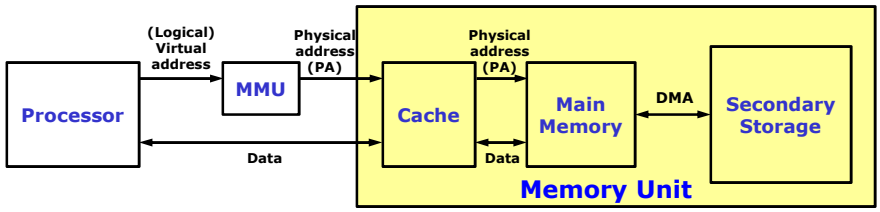
Virtual Memory Technique

- Technique that automatically move program and data blocks into the physical main memory when they are required for execution
- Using virtual program concept, each program may use entire CPU logical address space, at least up to secondary storage
- The address issued by the processor either for instruction or data are called virtual address or logical address
- These addresses are translated into physical memory addresses by a combination of hardware and software

26

Memory Management Unit (MMU)

- MMU translate the logical address into physical main memory address
- It is a part of the processor



- If the data is not in main memory, MMU causes the operating system to bring program or data into memory from the disk
- Transfer of data between disk and main memory is performed using direct memory access (DMA) scheme

27

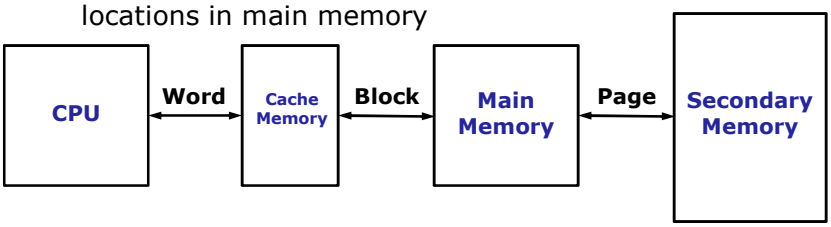
Page

- The programs or data in the disk are seen by the virtual memory as a collection of pages
- This page is the basic unit of information that is moved between the main memory and the secondary memory
- Each page is of the size 2 KB to 16 KB
- Page should not be too small
  - Disk access time is much longer
  - It take considerable time to locate data in the disk
- Page should not be too large
  - Substantial portion of a page may not be used
- Demand paging: Pages are copied to main memory when requested

28

Address Translation

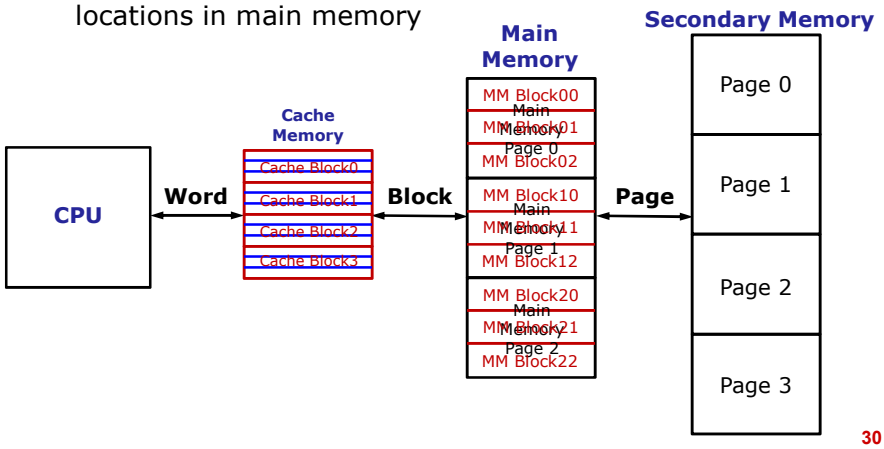
- The virtual memory address translation method is based on the concept of fixed length pages
- The address translations assumes that programs and data are composed of fixed size units called pages
- Unit of transfer between secondary memory and main memory is page
  - A page is a block of words that occupy contiguous locations in main memory



29

Address Translation

- The address translations assumes that programs and data are composed of fixed size units called pages
- Unit of transfer between secondary memory and main memory is page
  - A page is a block of words that occupy contiguous locations in main memory



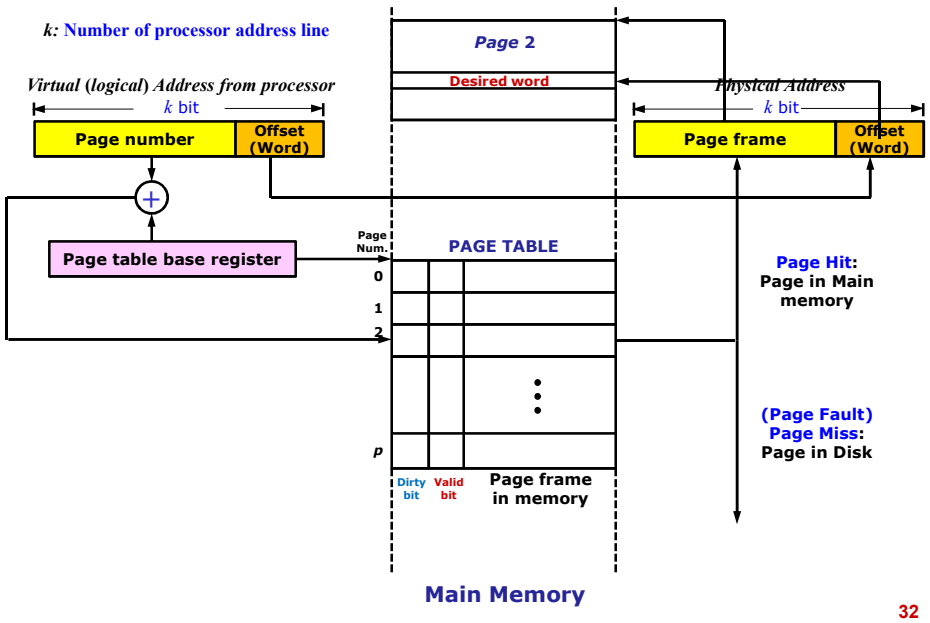
30

Parallels Between the Concepts of Cache and Virtual Memory

- Cache:
  - Bridges the speed gap between the processor and the main memory
  - It is implemented in hardware
- Virtual memory mechanism:
  - Bridges the size and speed gap between the main memory and secondary storage
  - It is usually implemented in part by software techniques
- Conceptually, cache techniques and main memory techniques are very similar
- They differ mainly in the details of their implementation

31

Virtual Memory Address Translation



32

Virtual Memory Address Translation

- The virtual address generated by the processor contain page number and offset (word) in the page
- To make sure that required page is in main memory, operating system create **page table for each process**
- This page table is kept in main memory
- Page table base register: Operating system keep the starting address of page table in it
- Page table hold the main memory location for each page
  - The area in main memory that can hold one page is called page frame
- Every entry in page table also include valid bit and dirty bit to describe the status of the page while it is in main memory

33

Virtual Memory Address Translation using Translation Lookaside Buffer

- Page table information is used by the MMU for every read and write access
- In order to speed up the address translation procedure, a small cache called Translation Lookaside Buffer (TLB) is incorporated in MMU
- It uses associative/set-associative mapping technique
- It hold a portion of page table corresponding to most recently accessed pages
- TLB holds only the page number and page frame number

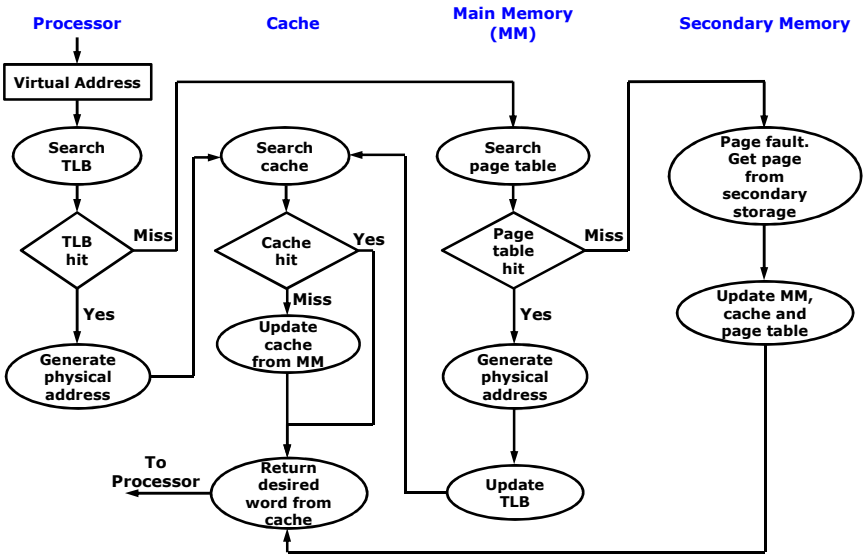
34

Virtual Memory Address Translation

- Page table information is used by the MMU for every read and write access
- Page fault: Whenever a requested page is not present in the main memory, page fault is said to have occurred
- When a page fault occurs, MMU asks operating system to intervene and raise an exception (interrupt)
  - Process in active get interrupted and control goes to operating system
  - Operating system then copies the requested page from disk to main memory
  - Then returns the control to the interrupted task
- During write operation pages get modified are indicated by dirty bit
- Modified page has to be written back to disk before removed from main memory
- Uses write back policy only

35

Operation of Memory Hierarchy and Virtual Memory Technique



36

Reference

- Hamacher, Zaky, "Computer Organization", 4th Edition
  - CH1, CH5 – 5.5 to 5.8

37