COMPUTER ORGANIZATION AND ARCHIECTURE

PROCESSOR STRUCTURE AND FUNCTION

D.MEENAKSHI
ASSISTANT PROFESSOR
DEPARTMENT OF IT&APPLICATIONS

POWER POINT PRESENTATION

II B.SC INFORMATION TECHNOLOGY

UNIT-IV

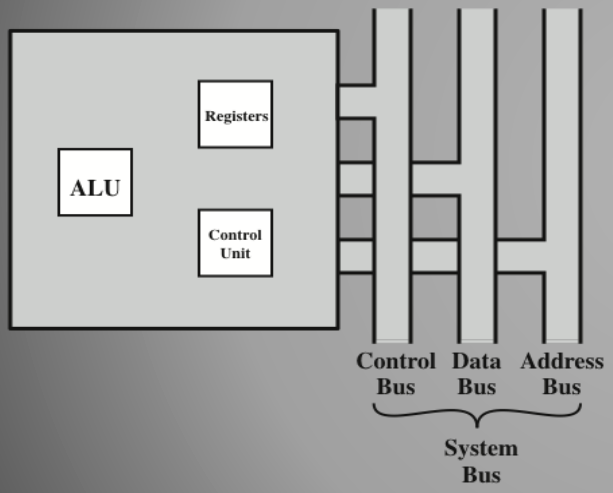
CONTENT

- Processor Organization
- Register Organization
- Instruction Cycle
- Control of the Processor

PROCESSOR ORGANIZATION

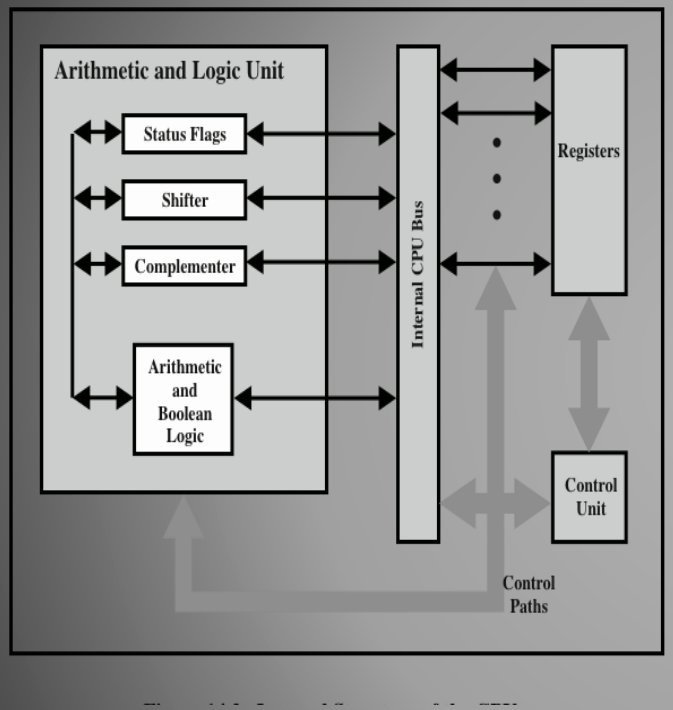
- **Fetch instruction:** The processor reads an instruction from memory
- **Interpret instruction**: The instruction is decoded and required action take place
- **Fetch data**: Execution of instruction where data read from memory/IO devices
- **Process data:** Perform arithmetic & Logical operations
- **Write data**: After the execution results are write into memory or I/O devices

CPU WITH SYSTEM BUS



- Major components of the processor ALU Control unit(CU) & Registers
- Connecting via system bus
- ALU: Perform Arithmetic and Logical operations
- CU: control the movement of data and instruction& control the operation of ALU
- Registers: set of storage locations

CPU INTERNAL ARCHITECTURE



- The data transfer and logic control paths labeled as internal processor bus
- Data transfer between various registers and ALU
- Major elements connected by data paths

REGISTER ORGANIZATION

- Processor have set of registers
- Registers function as a memory
- Registers in processor perform two roles

    (i) user visible registers

    (ii) control and status registers

USER VISIBLE REGISTERS

- **General purpose**

  Can be assigned to a variety of functions by the programmer

- **Data**

  May be used only to hold data and cannot be employed in the calculation of an operand address

- **Address**

  May be somewhat general purpose or may be devoted to a particular addressing mode

  Examples: segment pointers, index registers, stack pointer

- **Condition codes**

  Also referred to as *flags*

  Bits set by the processor hardware as the result of operations

CONDITION CODES

Advantages	Disadvantages
1. Because condition codes are set by normal arithmetic and data movement instructions, they should reduce the number of COMPARE and TEST instructions needed. 2. Conditional instructions, such as BRANCH are simplified relative to composite instructions, such as TEST AND BRANCH. 3. Condition codes facilitate multiway branches. For example, a TEST instruction can be followed by two branches, one on less than or equal to zero and one on greater than zero. 4. Condition codes can be saved on the stack during subroutine calls along with other register information.	1. Condition codes add complexity, both to the hardware and software. Condition code bits are often modified in different ways by different instructions, making life more difficult for both the microprogrammer and compiler writer. 2. Condition codes are irregular; they are typically not part of the main data path, so they require extra hardware connections. 3. Often condition code machines must add special non-condition-code instructions for special situations anyway, such as bit checking, loop control, and atomic semaphore operations. 4. In a pipelined implementation, condition codes require special synchronization to avoid conflicts.

CONTROL AND STATUS REGISTERS

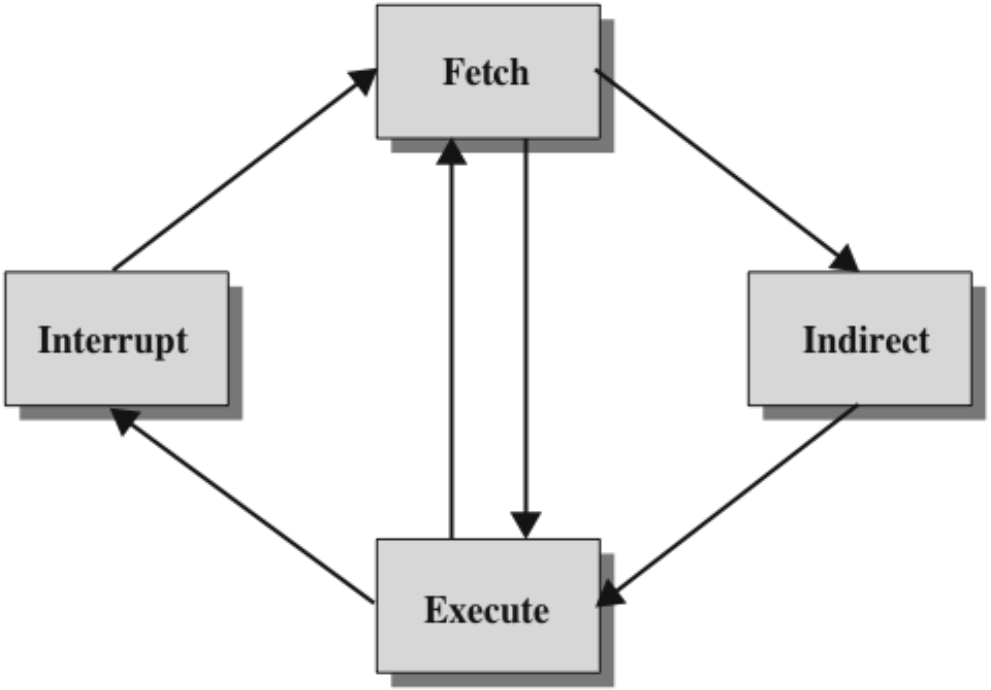
- Control the operation of the processor
- **Program counter(PC):** Next address of the instruction to be fetched
- **Instruction register(IR):** Instruction most recently fetched
- **Memory address register(MAR):** The address of location in memory
- **Memory buffer register(MBR):** A word of data to be written to memory

PROGRAM STATUS WORD

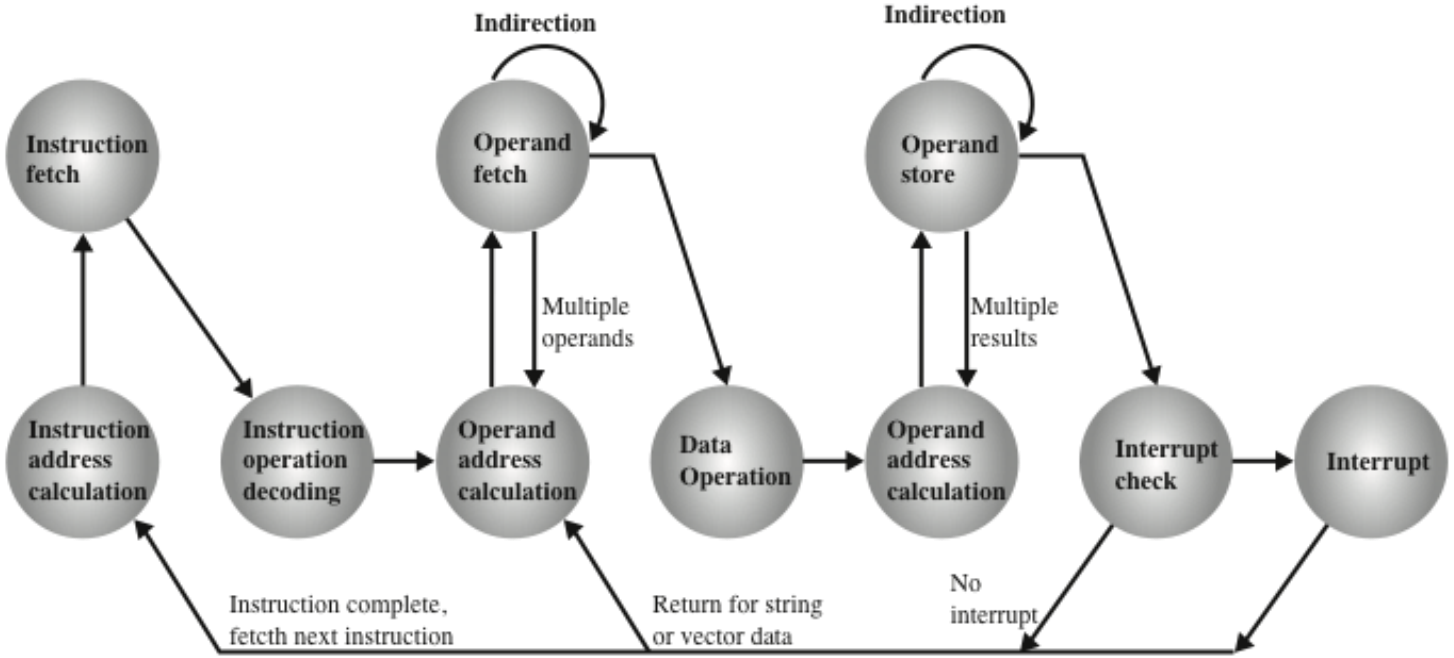
- Many processor designs a set of registers that contain status information
- **PSW :**Contains condition codes plus other status information
- **Sign :** Contains sign bit of the result of arithmetic operation
- **Zero:** Set when result is 0
- **Carry :**Set if an operation resulted in a carry into or borrow out of a high order bit
- **Equal :**Set if a logical compare result is equality
- **Overflow :** Used to indicate arithmetic overflow
- **Interrupt enable/disable:** Used to enable or disable interrupts
- **Supervisor :** Indicates whether the processor is executing in supervisor or user mode

INSTRUCTION CYCLE

- **Fetch:** Read the next instruction from memory in to processor
- **Execute:** Interpret the opcode and perform the indicated operation
- **Interrupt:** If interrupts are enabled and an interrupt has occurred, save the current process state in to the stack and service the interrupt
- Indirect cycle: after an instruction is fecthed it is examined to determine if any indirect addressing is involved. The required operands are fetched using indirect addressing.

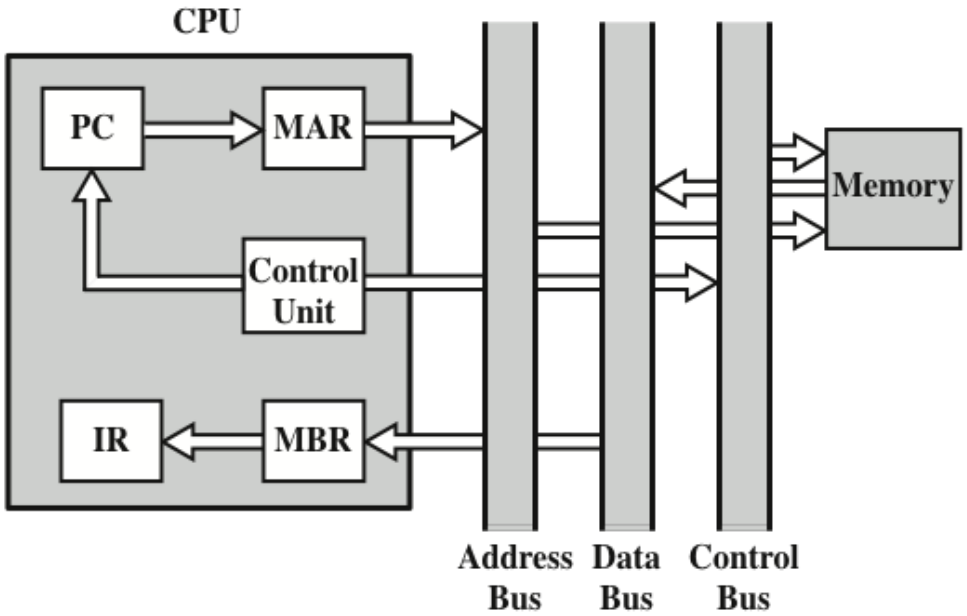

Fetch
Interrupt
Indirect
Execute

INSTRUCTION CYCLE STATE DIAGRAM



DATA FLOW FETCH CYCLE

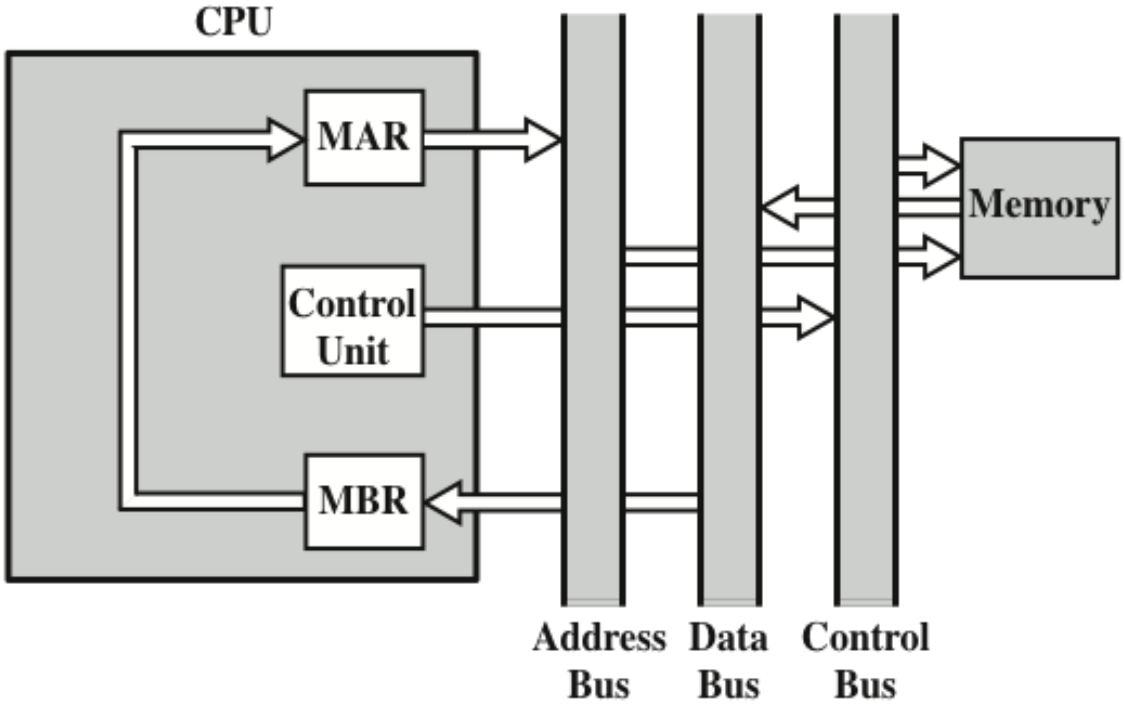
- During fetch cycle an instruction is read from memory.
- The PC contains the address of the next instruction to be fetched.
- This address is moved to MAR
- The result is placed on data bus and copy into MBR and moved to IR.



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

DATA FLOW INDIRECT CYCLE

- Fetch cycle is over, the control unit examine the content of IR to determine if it contain an operands specify using indirect addressing.
- Indirect cycle is performed.
- The fetch and indirect cycles are simple and predictable.


CPU
MAR
Control Unit
MBR
Memory
Address Bus
Data Bus
Control Bus

DATA FLOW INTERRUPT CYCLE

- The current content of PC saved to the processor resume normal activities after the interrupt.
- The content of PC are transfer to MBR to be written into memory.
- PC is loaded with the address of the interrupt routine.

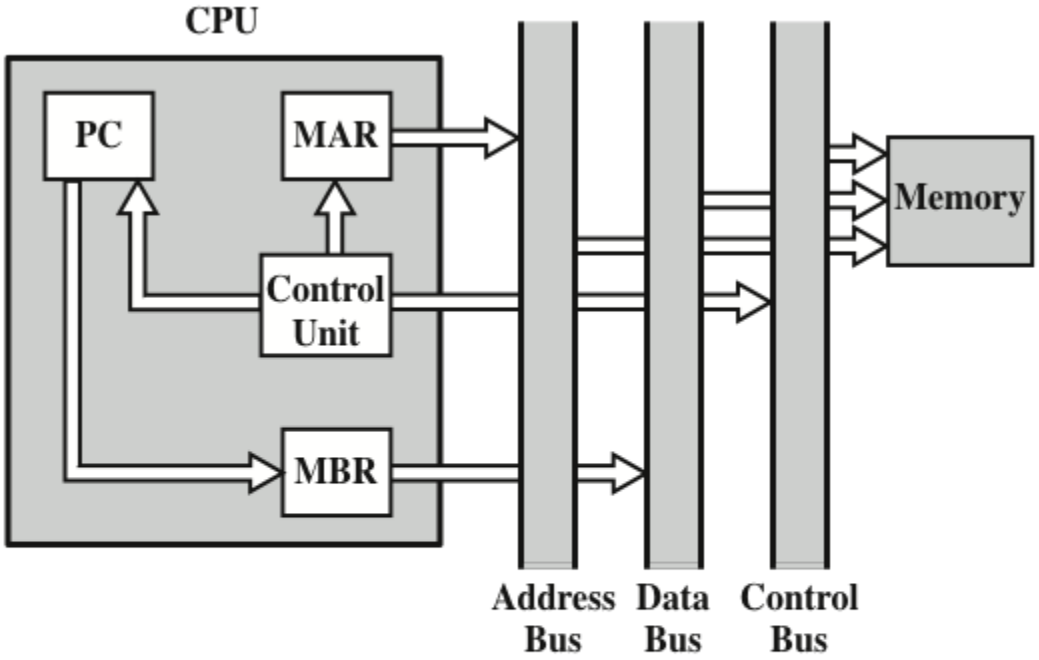


Figure 14.8 Data Flow, Interrupt Cycle

Functional Requirements

- Define basic elements of processor
- Describe micro-operations processor performs
- Determine functions control unit must perform

Basic Elements of Processor

- ALU
- Registers
- Internal data paths
- External data paths
- Control Unit

Types of Micro-operation

- Transfer data between registers
- Transfer data from register to external
- Transfer data from external to register
- Perform arithmetic or logical ops

Control Signals

- Inputs of control signals:
- Clock

  one micro-instruction (or set of parallel micro-instructions) per clock cycle
- Instruction register

  Op-code for current instruction

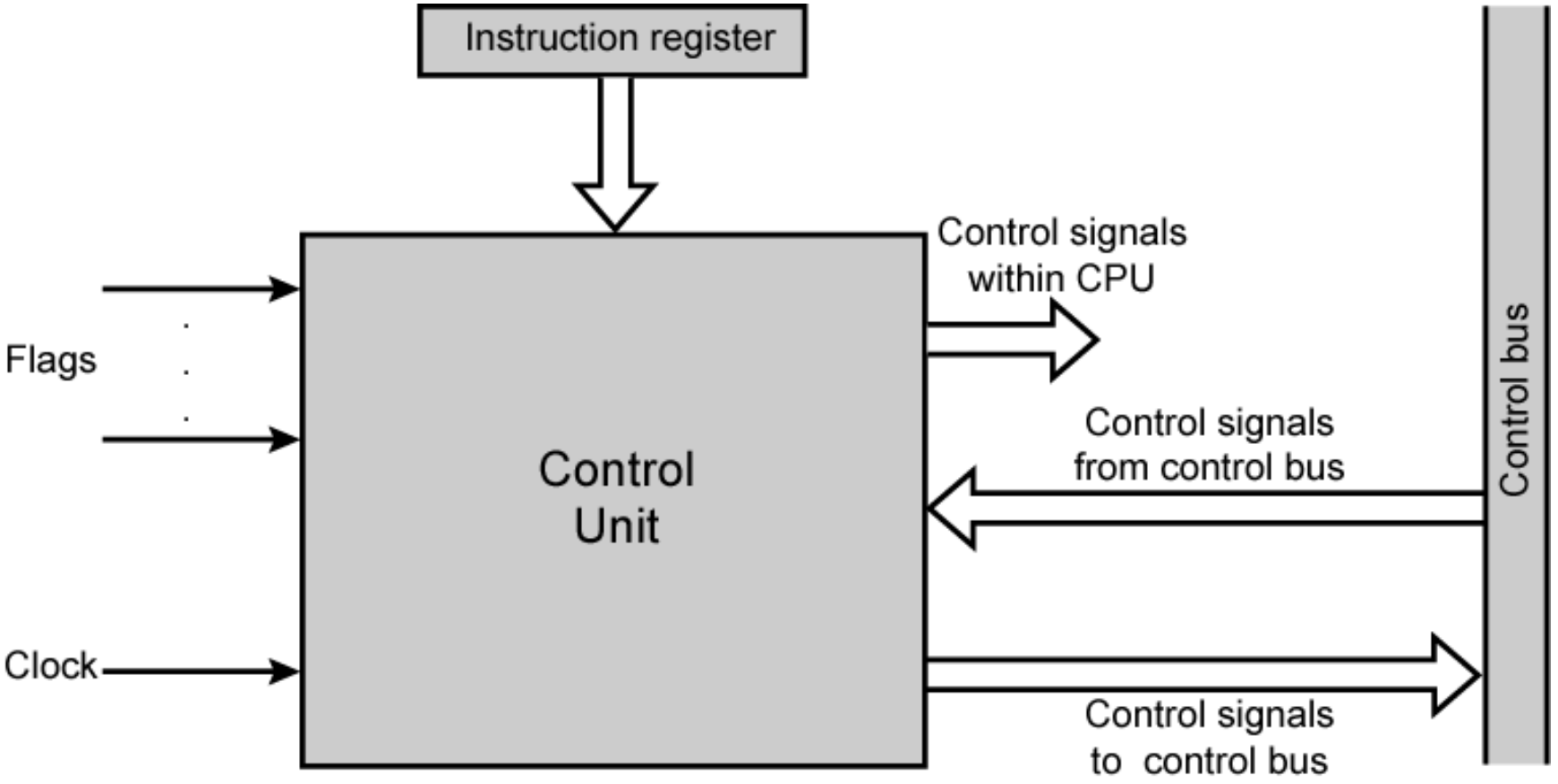
  Determines which micro-instructions are performed
- Flags

  State of CPU

  Results of previous operations
- From control bus

  Interrupts

  Acknowledgements


Instruction register
Control signals
within CPU
Flags
Control
Unit
Control signals
from control bus
Control bus
Clock
Control signals
to control bus

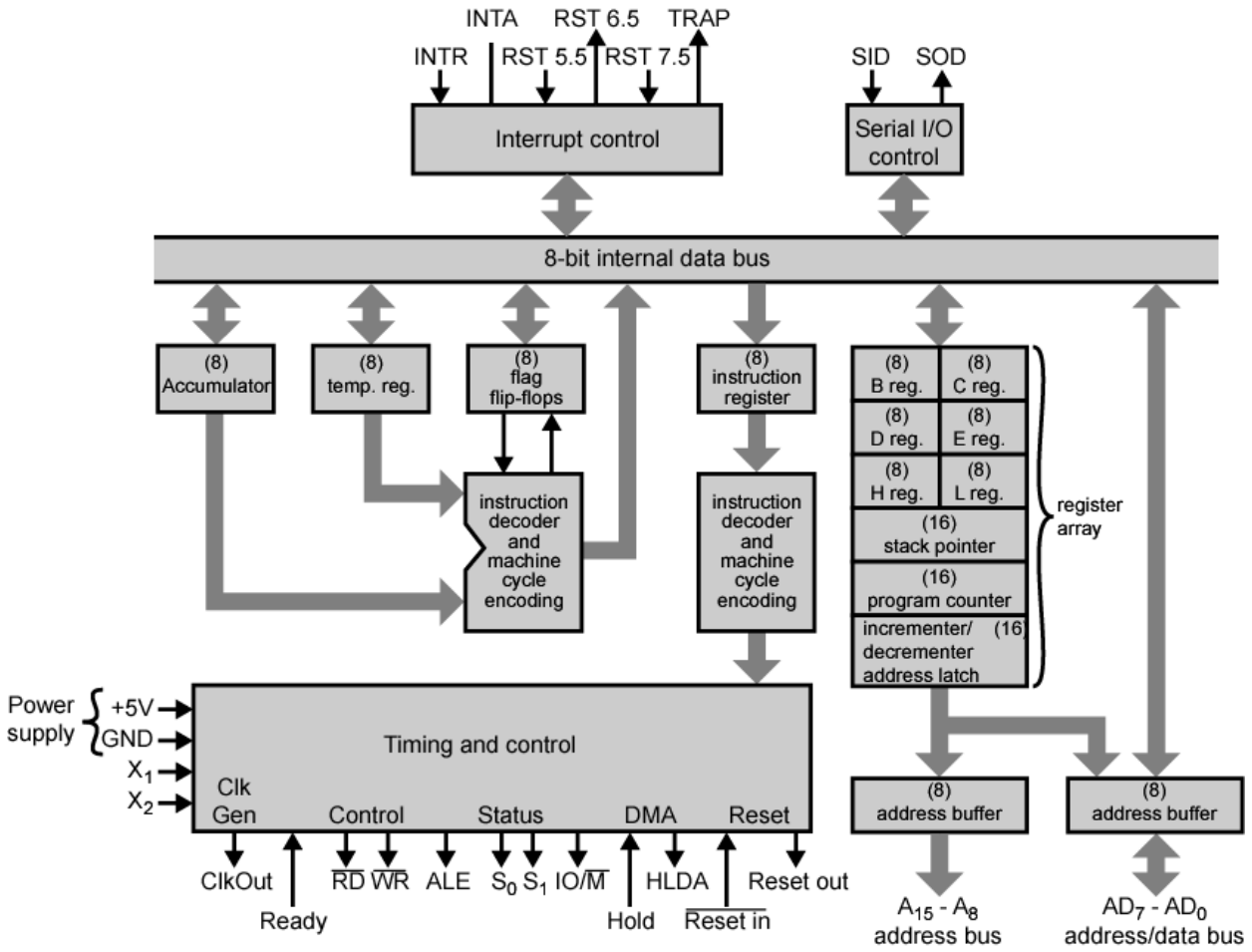
Control signals

- Output of control signals
- Within CPU
  - Cause data movement
  - Activate specific functions
- Via control bus
  - To memory
  - To I/O modules

Example of the processor(8085)

- Intel 8085 is an 8-bit, NMOS microprocessor.
- It is a 40 pin C package fabricated on a single LSI chip.
- The Intel 8085A uses a single +5V D.C supply for its operation.
- It has 80 basic instructions and 246 opcodes.
- It consists of three main sections, an arithmetic and logic unit a timing and control unit and several registers.

ARCHITECTURE DIAGRAM 8085



The typical processor system consists of:

- CPU (central processing unit)
- ALU (arithmetic-logic unit)
- Control Logic
- Registers, etc…
- Memory
- Input / Output interfaces

Interconnections between these units:

- Address Bus
- Data Bus
- Control Bus

The internal architecture of the 8085 CPU is capable of performing the following operations:

- Store 8-bit data (Registers, Accumulator)
- Perform arithmetic and logic operations (ALU)
- Test for conditions (IF / THEN)
- Sequence the execution of instructions
- Store temporary data in RAM during execution

- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address bits are multiplexed (time shared) with the 8 data bits.

So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.

During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.

In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

Registers

- Six general purpose 8-bit registers: B, C, D, E, H, L
- They can also be combined as register pairs to perform 16-bit operations: BC, DE, HL
- Registers are programmable (data load, move, etc.)

Accumulator

- Single 8-bit register that is part of the ALU !
- Used for arithmetic / logic operations – the result is always stored in the accumulator.

- The Program Counter (PC)

  This is a register that is used to control the sequencing of the execution of instructions.

  This register always holds the address of the next instruction.

  Since it holds an address, it must be 16 bits wide.

- The Stack pointer

  The stack pointer is also a 16-bit register that is used to point into memory.

  The memory this register points to is a special area called the stack.

  The stack is an area of memory used to hold data that will be retreived soon.

  The stack is usually accessed in a Last In First Out (LIFO) fashion.

THANK YOU