Computer Architecture

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr
nizamettinaydin@gmail.com

http://www.yildiz.edu.tr/~naydin

1

Computer Architecture

Control Unit Operation

2

Outline

- Control Unit Operation
  - Major Advances in Computers
  - Micro-Operations
  - Constituent Elements of Program Execution
  - Basic Functional Elements of Processor
  - Types of Micro-operation
  - Functions of Control Unit
  - Control Signals
  - Internal Processor Organization

3

Major Advances in Computers(1)

- The family concept
  - IBM System/360 1964
  - DEC PDP-8
  - Separates architecture from implementation
- Microprogrammed control unit
  - Idea by Wilkes 1951
  - Produced by IBM S/360 1964
- Cache memory
  - IBM S/360 model 85 1969

4

Major Advances in Computers(2)

- Solid State RAM
- Microprocessors
  - Intel 4004 1971
- Pipelining
  - Introduces parallelism into fetch execute cycle
- Multiple processors

5

Functional requirements for a processor

The following list determines what a processor must do:

- Operations (opcodes)
- Addressing modes
- Registers
- I/O module interfaces
- Memory module interfaces
- Interrupt processing structure

Question is:

how the verious elements of the processor are controlled to provide these functions?

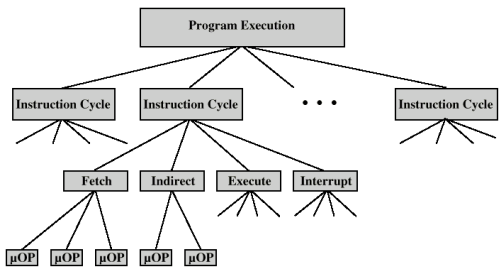
6

Micro-Operations

- A computer executes a program
  - Program cycle
- A program is a collection of
  - instruction cycles
- An instruction cycle is made of
  - Fetch and execute cycles
- Each cycle has a number of steps
  - Called micro-operations
- Each step does very little
  - Atomic operation of CPU

7

Constituent Elements of Program Execution



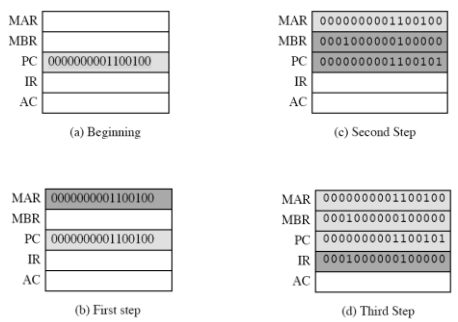
8

Fetch Cycle – (relevant Registers)

- Memory Address Register (MAR)
  - Connected to address bus
  - Specifies address for read or write op
- Memory Buffer Register (MBR)
  - Connected to data bus
  - Holds data to write or last data read
- Program Counter (PC)
  - Holds address of next instruction to be fetched
- Instruction Register (IR)
  - Holds last instruction fetched

9

Sequence of events, Fetch cycle



10

Fetch Sequence

- Address of next instruction is in PC
- Address (MAR) is placed on address bus
- Control unit issues READ command
- Result (data from memory) appears on data bus
- Data from data bus copied into MBR
- PC incremented by 1 (in parallel with data fetch from memory)
- Data (instruction) moved from MBR to IR
- MBR is now free for further data fetches

11

Fetch Sequence (symbolic)

- t1: MAR <- (PC)
- t2: MBR <- (memory)
  PC <- (PC) +1
- t3: IR <- (MBR)
  (tx = time unit/clock cycle)

or

- t1: MAR <- (PC)
- t2: MBR <- (memory)
- t3: PC <- (PC) +1
  IR <- (MBR)

12

Rules for Clock Cycle Grouping

- Proper sequence must be followed
  - MAR <- (PC) must precede MBR <- (memory)
- Conflicts must be avoided
  - Must not read & write same register at same time
  - MBR <- (memory) & IR <- (MBR) must not be in same cycle
- Also: PC <- (PC) +1 involves addition
  - Use ALU
  - May need additional micro-operations

13

Indirect Cycle

- MAR <- ($IR_{address}$) - address field of IR
- MBR <- (memory)
- $IR_{address}$ <- ($MBR_{address}$)

- MBR contains an address
- IR is now in same state as if direct addressing had been used
- (What does this say about IR size?)

14

Interrupt Cycle

- t1: MBR <-(PC)
- t2: MAR <- save-address
  PC <- routine-address
- t3: memory <- (MBR)

- This is a minimum
  - May be additional micro-ops to get addresses

15

Execute Cycle (ADD)

- Different for each instruction
  - e.g. ADD R1,X - add the contents of location X to Register 1 , result in R1

- t1: MAR <- ($IR_{address}$)
- t2: MBR <- (memory)
- t3: R1 <- R1 + (MBR)

- Note no overlap of micro-operations

16

Execute Cycle (ISZ)

- ISZ X - increment and skip if zero

- t1: MAR <- ($IR_{address}$)
- t2: MBR <- (memory)
- t3: MBR <- (MBR) + 1
- t4: memory <- (MBR)
  if (MBR) == 0 then PC <- (PC) + 1

17

Execute Cycle (BSA)

- BSA X - Branch and save address
  - Address of instruction following BSA is saved in X
  - Execution continues from X+1

- t1: MAR <- ($IR_{address}$)
  MBR <- (PC)
- t2: PC <- ($IR_{address}$)
  memory <- (MBR)
- t3: PC <- (PC) + 1

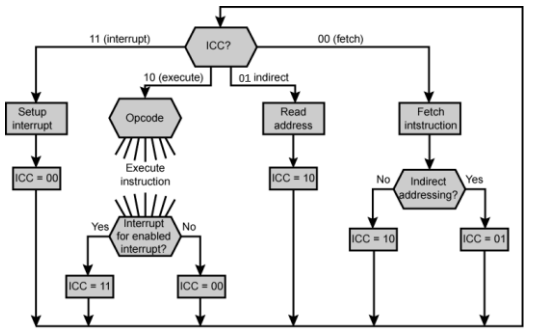
18

Instruction Cycle

- Each phase decomposed into sequence of elementary micro-operations
- E.g. fetch, indirect, and interrupt cycles
- Execute cycle
  - One sequence of micro-operations for each opcode
- Need to tie sequences together
- Assume new 2-bit register
  - Instruction cycle code (ICC) designates which part of cycle processor is in
    - 00: Fetch
    - 01: Indirect
    - 10: Execute
    - 11: Interrupt

19

Flowchart for Instruction Cycle



20

Basic Funcional Elements of Processor

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

21

Types of Micro-operation

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

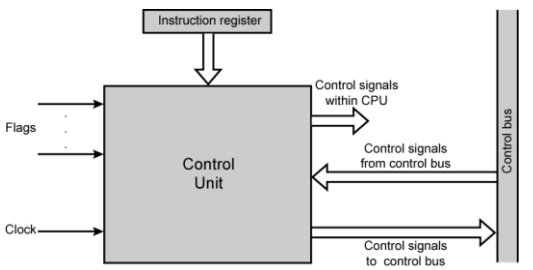
22

Functions of Control Unit

- Sequencing
  - Causing the CPU to step through a series of micro-operations
- Execution
  - Causing the performance of each micro-op
- This is done using Control Signals

23

Model of Control Unit



24

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

25

Control Signals - output

- Control signals within CPU
  - Cause data movement
  - Activate specific functions
- Control signals to control bus
  - To memory
  - To I/O modules

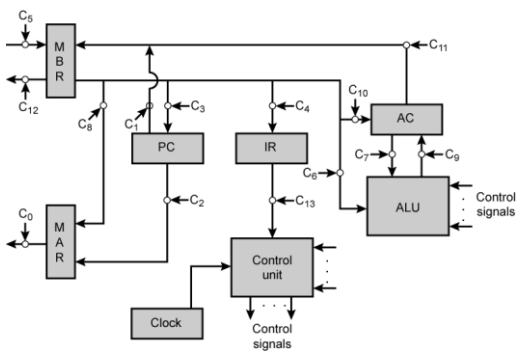
26

Example Control Signal Sequence-Fetch

- MAR <- (PC)
  - Control unit activates signal to open gates between PC and MAR
- MBR <- (memory)
  - Open gates between MAR and address bus
  - Memory read control signal
  - Open gates between data bus and MBR

27

Data Paths and Control Signals



28

Micro-operations and Control Signals

Micro-operations	Timing	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.
C_W = Write control signal to system bus.

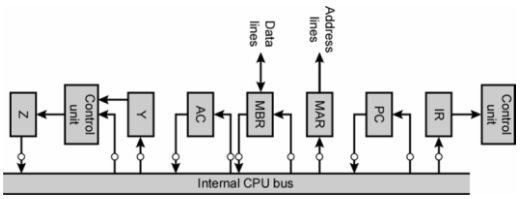
29

Internal Processor Organization

- Usually a single internal bus
- Gates control movement of data onto and off the bus
- Control signals control data transfer to and from external systems bus
- Temporary registers needed for proper operation of ALU

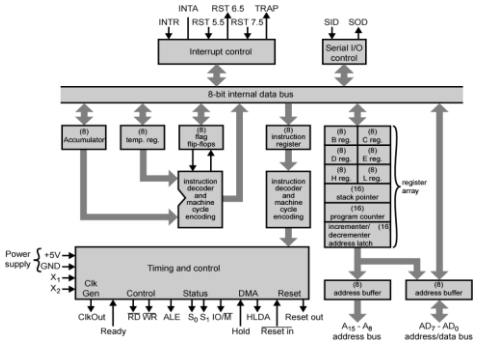
30

CPU with Internal Bus



31

Intel 8085 CPU Block Diagram



32

Intel 8085 External Signals...

Address and Data Signals

High Address (A15–A8)
The high order 8 bits of a 16-bit address.

Address/Data (AD7–AD0)
The lower-order 8 bits of a 16-bit address or 8 bits of data. This multiplexing saves on pins.

Serial Input Data (SID)
A single-bit input to accommodate devices that transmit serially (one bit at a time).

Serial Output Data (SOD)
A single-bit output to accommodate devices that receive serially.

Timing and Control Signals

CLK (OUT)
The system clock. Each cycle represents one T state. The CLK signal goes to peripheral chips and synchronizes their timing.

X1, X2
These signals come from an external crystal or other device to drive the internal clock generator.

Address Latch Enabled (ALE)
Occurs during the first clock state of a machine cycle and causes peripheral chips to store the address lines. This allows the address module (e.g., memory, I/O) to recognize that it is being addressed.

Status (S0, S1)
Control signals used to indicate whether a read or write operation is taking place.

IO/M
Used to enable either I/O or memory modules for read and write operations.

Read Control (RD)
Indicates that the selected memory or I/O module is to be read and that the data bus is available for data transfer.

Write Control (WR)
Indicates that data on the data bus is to be written into the selected memory or I/O location.

33

Intel 8085 External Signals

Memory and I/O Initiated Symbols

Hold
Requests the CPU to relinquish control and use of the external system bus. The CPU will complete execution of the instruction presently in the IR and then enter a hold state, during which no signals are inserted by the CPU to the control, address, or data buses. During the hold state, the bus may be used for DMA operations.

Hold Acknowledge (HOLDA)
This control unit output signal acknowledges the HOLD signal and indicates that the bus is now available.

READY
Used to synchronize the CPU with slower memory or I/O devices. When an addressed device asserts READY, the CPU may proceed with an input (DBIN) or output (WR) operation. Otherwise, the CPU enters a wait state until the device is ready.

Interrupt-Related Signals

TRAP
Restart Interrupts (RST 7.5, 6.5, 5.5)

Interrupt Request (INTR)
These five lines are used by an external device to interrupt the CPU. The CPU will not honor the request if it is in the hold state or if the interrupt is disabled. An interrupt is honored only at the completion of an instruction. The interrupts are in descending order of priority.

Interrupt Acknowledge
Acknowledges an interrupt.

CPU Initialization

RESET IN
Causes the contents of the PC to be set to zero. The CPU resumes execution at location zero.

RESET OUT
Acknowledges that the CPU has been reset. The signal can be used to reset the rest of the system.

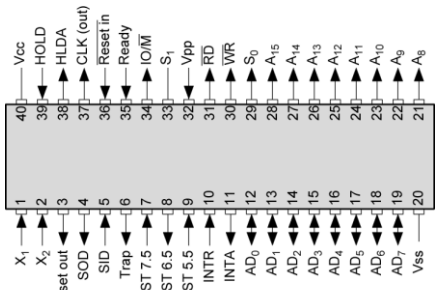
Voltage and Ground

VCC
+5 volt power supply

VSS
Electrical ground.

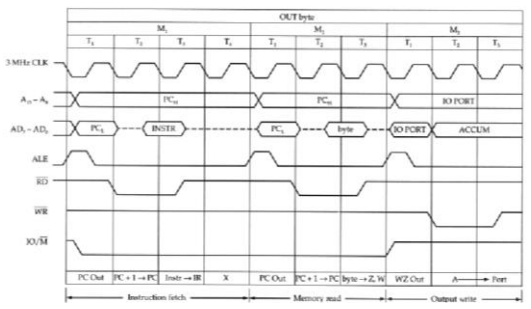
34

Intel 8085 Pin Configuration



35

Intel 8085 OUT Instruction Timing Diagram



36