

CSE 321a

Computer Organization (1)

تنظيم الحاسبات (1)



3rd year, Computer Engineering
Fall 2017

Lecture #10



Dr. Hazem Ibrahim Shehata

Dept. of Computer & Systems Engineering

Credits to Dr. Ahmed Abdul-Monem Ahmed for the slides

Administrivia

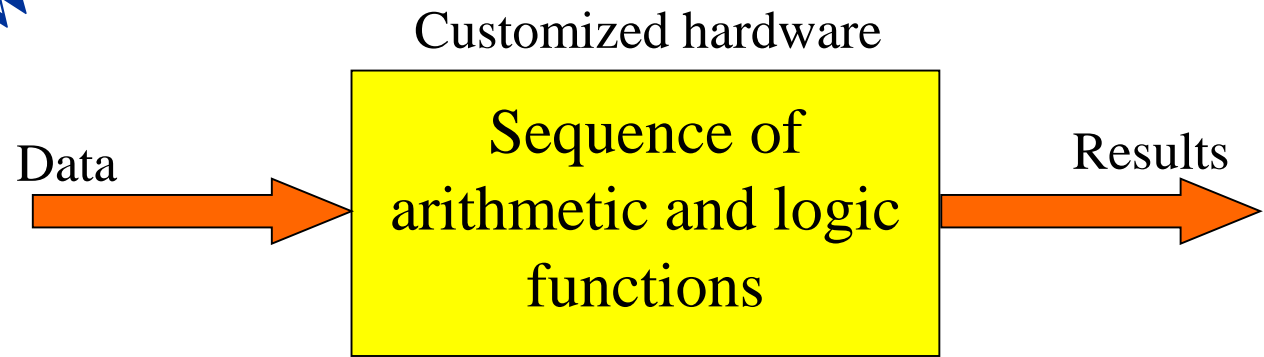
- Midterm:
 - Marks have been posted.
- Assignment #3:
 - To be released by next week.

Website: <http://hshehata.github.io/courses/zu/cse321a>

Office hours: Sunday 1:00pm-2:00pm

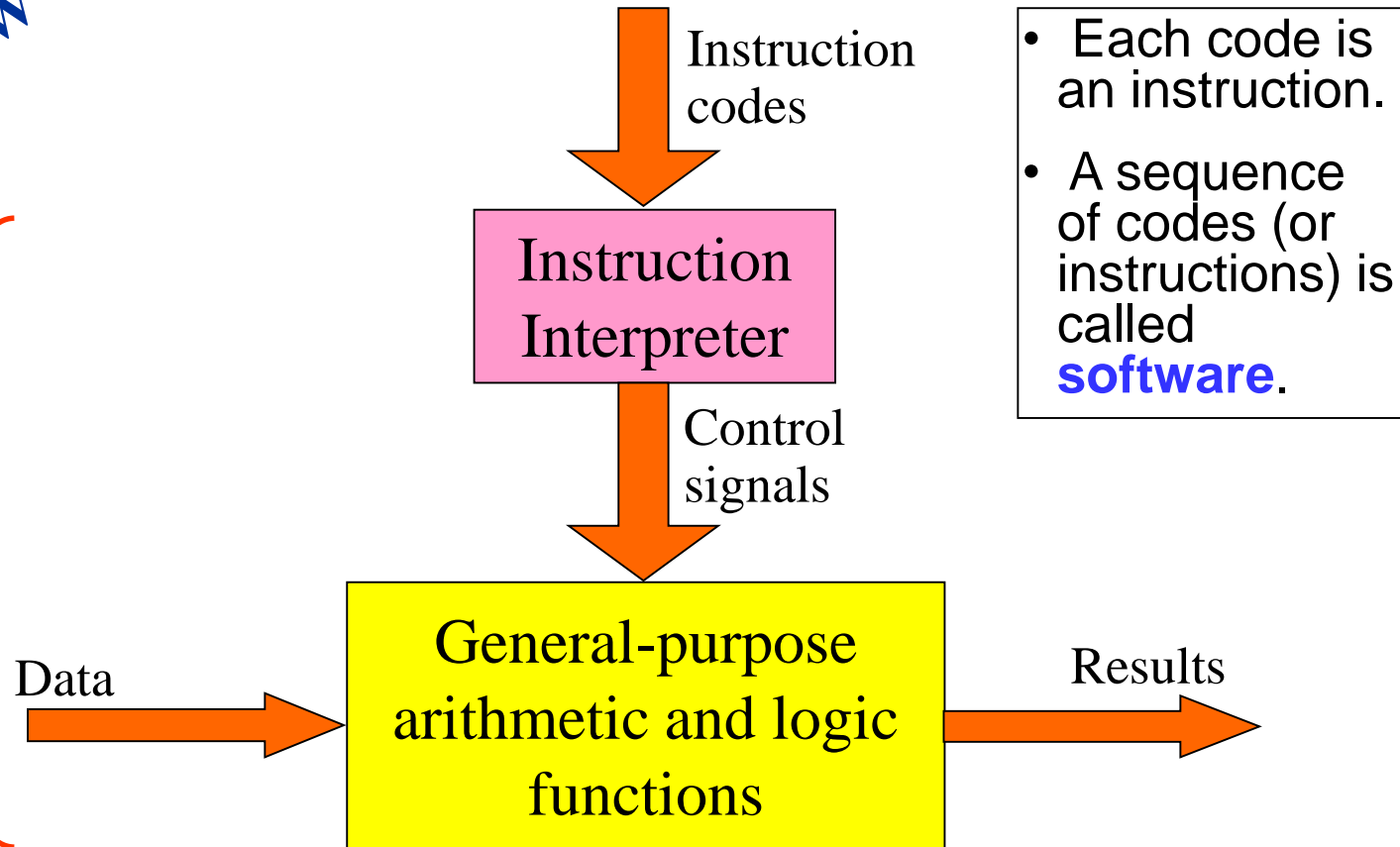
Chapter 19. Control Unit Operation (Online Appendix)

Programming in H/W



Programming in S/W

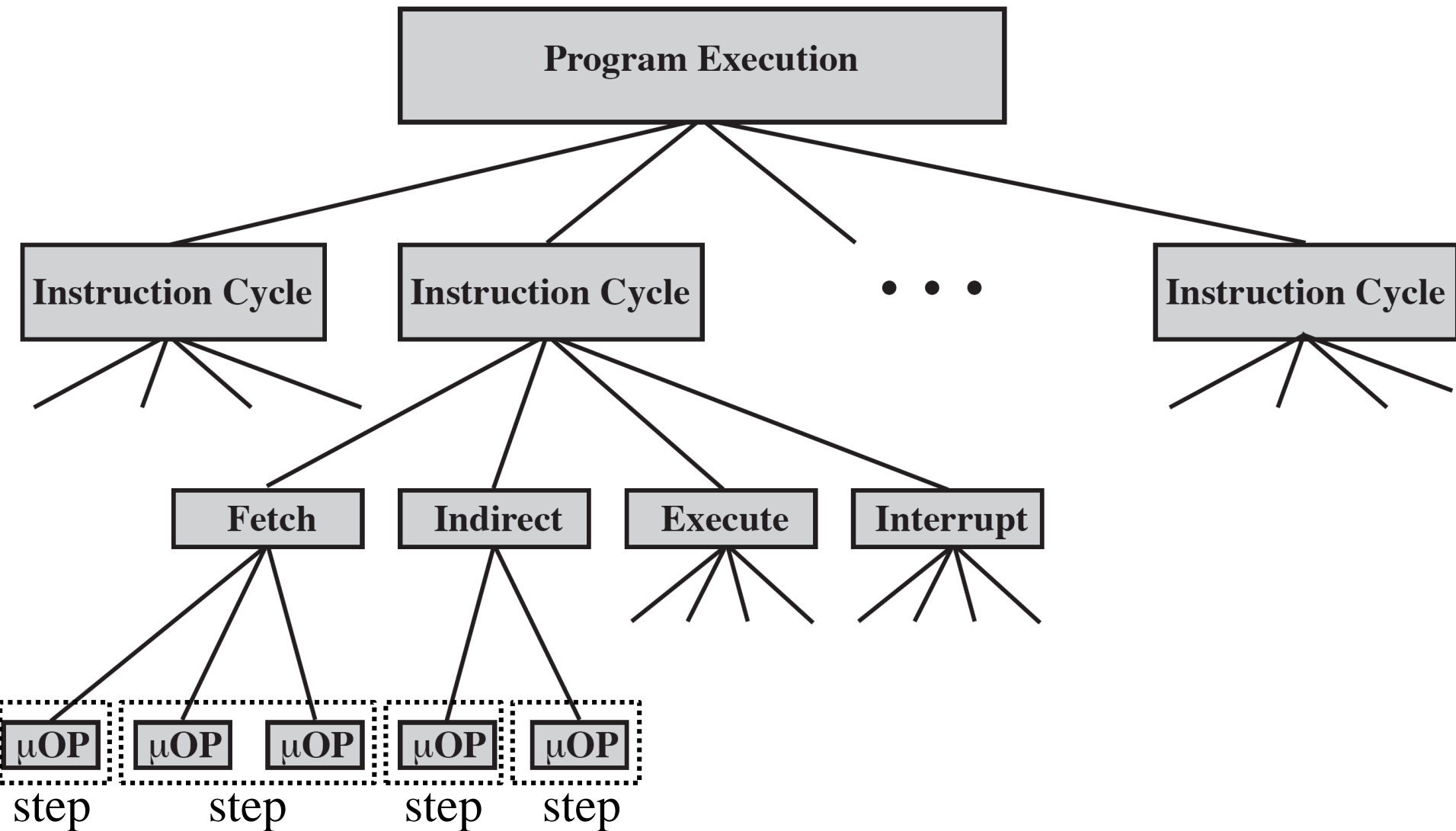
CPU



Micro-operations

- A computer executes **programs**.
- A program is a sequence of instructions → **instruction cycles**.
- Each instruction is made up of smaller units (**subcycles**).
 - e.g., fetch/execute subcycles.
- Each subcycle has a number of **steps**.
- Each step has a number of **micro-operations**.
- Each micro-operation does very little.
- Micro-operation is an atomic operation of CPU.
- Any instruction is a sequence of micro-operations.

Constituent Elements of Prog. Execution



Fetch Subcycle in simple machine

MAR	0000000001100100
MBR	0001000000100000
PC	00000000011001000
IR	0001101001101000
AC	

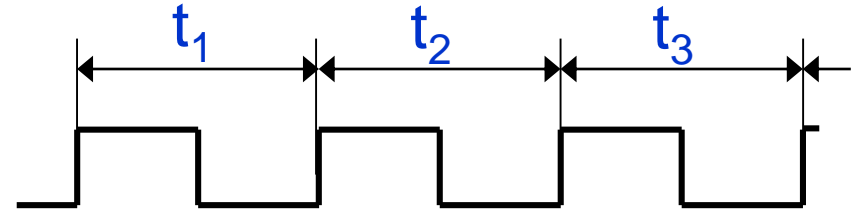
- **Memory Address Register (MAR)**
 - Specifies address for read or write operation.
 - Connected to address bus.
- **Memory Buffer Register (MBR)**
 - Holds data to write or last data read.
 - Connected to data bus.
- **Program Counter (PC)**
 - Holds address of next instruction to be fetched.
- **Instruction Register (IR)**
 - Holds last instruction fetched (being executed).
- **Accumulator (AC)**
 - Acts as implicit operand.

Fetch Sequence

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

Fetch Subcycle (symbolic)

- t_1 : $MAR \leftarrow [PC]$
- t_2 : $MBR \leftarrow \text{Memory},$
 $PC \leftarrow [PC] + I$
- t_3 : $IR \leftarrow [MBR]$ Micro-operation



(tx = time unit/clock cycle)

Micro-operations take equal times.

Or

- t_1 : $MAR \leftarrow [PC]$
- t_2 : $MBR \leftarrow \text{Memory}$
- t_3 : $PC \leftarrow [PC] + I,$
 $IR \leftarrow [MBR]$

Rules for Clock Cycle Grouping

- Proper sequence must be followed
 - $\text{MAR} \leftarrow [\text{PC}]$ must precede $\text{MBR} \leftarrow \text{Memory}$
- Read/write conflicts must be avoided
 - A μop reading from reg. X cannot be scheduled at the same step (cycle) with another μop writing to reg. X.
 - Ex.: $\text{MBR} \leftarrow \text{Memory}$ & $\text{IR} \leftarrow [\text{MBR}]$ must not be in same step.
- Resource conflicts must be avoided
 - For instance, 2 μop 's involving addition cannot be scheduled at the same step if ALU contains 1 adder only.
 - Note: $\text{PC} \leftarrow [\text{PC}] + 1$ involves addition, and hence use ALU.

Indirect Subcycle

- t_1 : $MAR \leftarrow [IR\text{-}address]$ - address field of IR
 - t_2 : $MBR \leftarrow Memory$
 - t_3 : $IR\text{-}address \leftarrow [MBR]$
-
- MBR contains an address.
 - IR is now in same state as if direct addressing had been used.

Interrupt Subcycle

- At the end of the execute subcycle, interrupts are tested.
- Some interrupt occurred → interrupt subcycle.
- Assume: address of ISR routine & address of location to save PC become available in the second step.
- t_1 : $MBR \leftarrow [PC]$
- t_2 : $MAR \leftarrow \text{save-address}, PC \leftarrow \text{routine-address}$
- t_3 : $\text{Memory} \leftarrow [MBR]$
- This is a minimum
 - May be additional micro-ops to get addresses.
 - N.B. saving context is done by interrupt handler routine, not micro-ops.

Execute Subcycle (ADD)

- Execute subcycle is different for each instruction.
- ADD X
 - Add contents of location X to AC and save result to AC
- t_1 : $MAR \leftarrow [IR\text{-}address]$
- t_2 : $MBR \leftarrow Memory$
- t_3 : $AC \leftarrow [AC] + [MBR]$
- Note no overlap of micro-operations.

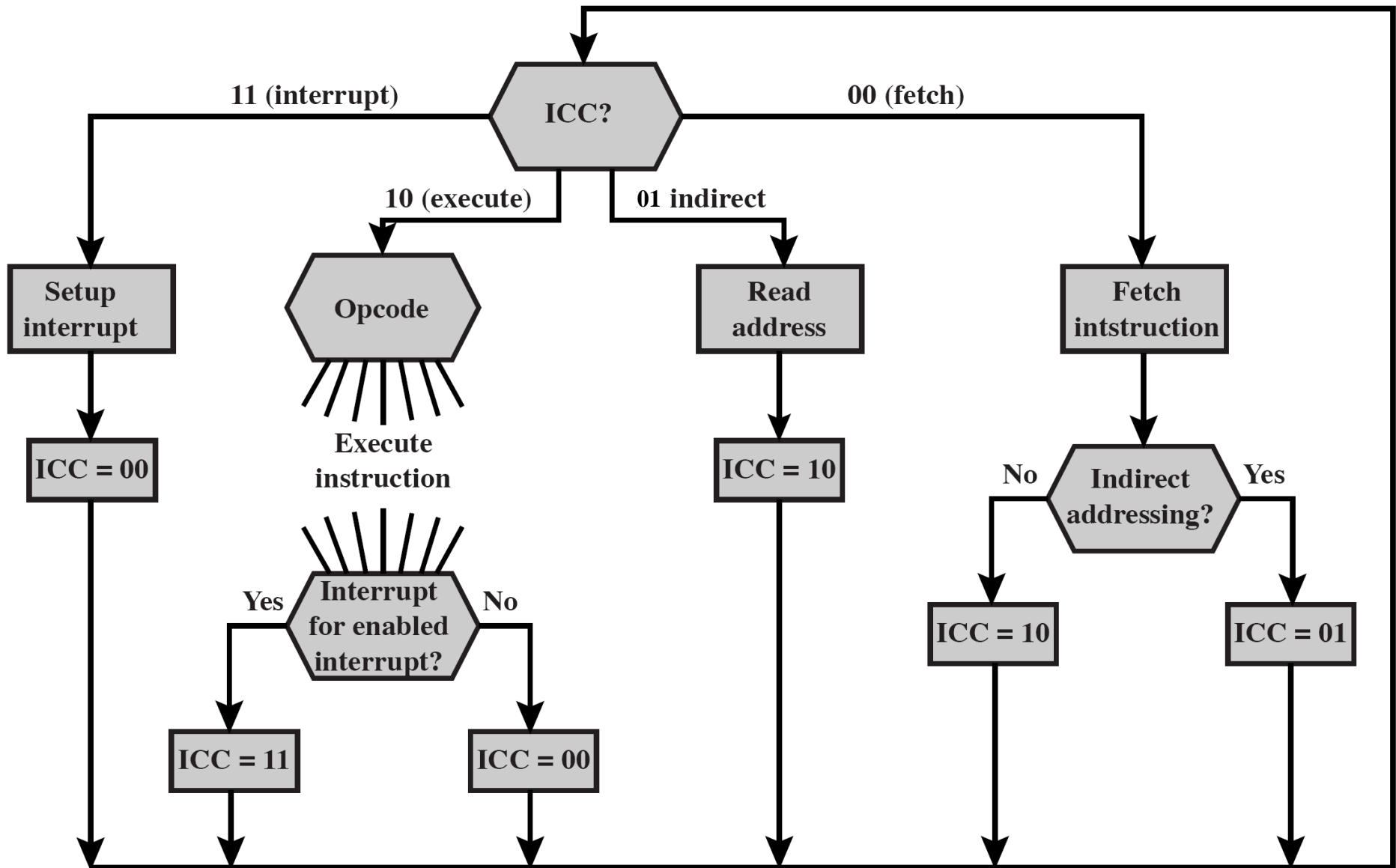
Execute Subcycle (ISZ)

- ISZ X - increment and skip if zero
 - t_1 : $MAR \leftarrow [IR\text{-}address]$
 - t_2 : $MBR \leftarrow Memory$
 - t_3 : $MBR \leftarrow [MBR] + 1$
 - t_4 : $Memory \leftarrow [MBR]$
if $[MBR] == 0$ then $PC \leftarrow [PC] + 1$
- Notes:
 - “If ...” is a single micro-operation.
 - Micro-operations done during t_4 .

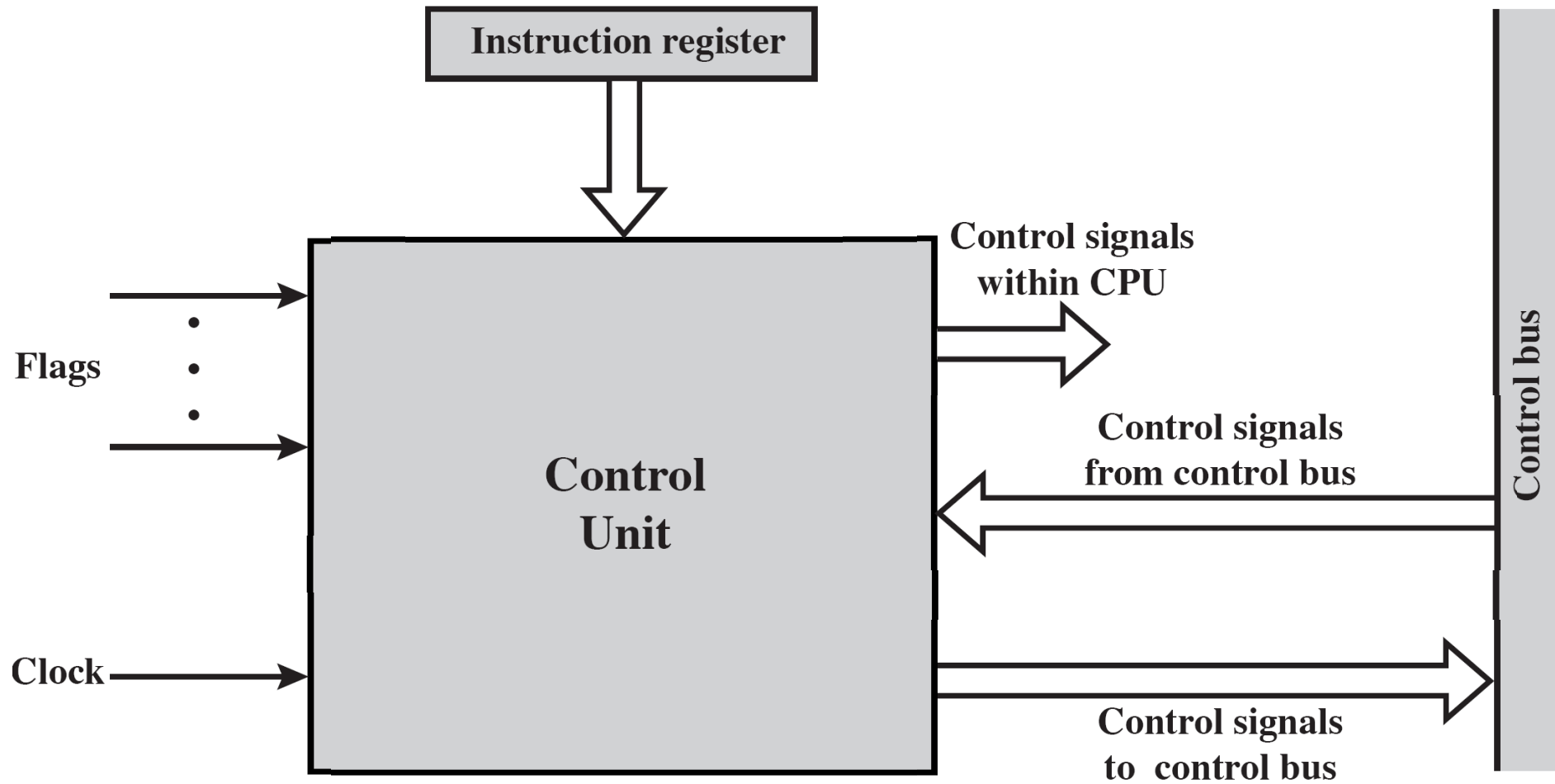
Execute Subcycle (BSA)

- BSA X - Branch and save address
 - Address of instruction following BSA is saved in X
 - Execution continues from X+1
- t_1 : $MAR \leftarrow [IR\text{-}address], MBR \leftarrow [PC]$
- t_2 : $PC \leftarrow [IR\text{-}address], Memory \leftarrow [MBR]$
- t_3 : $PC \leftarrow [PC] + 1$

Instruction Cycle – Flowchart



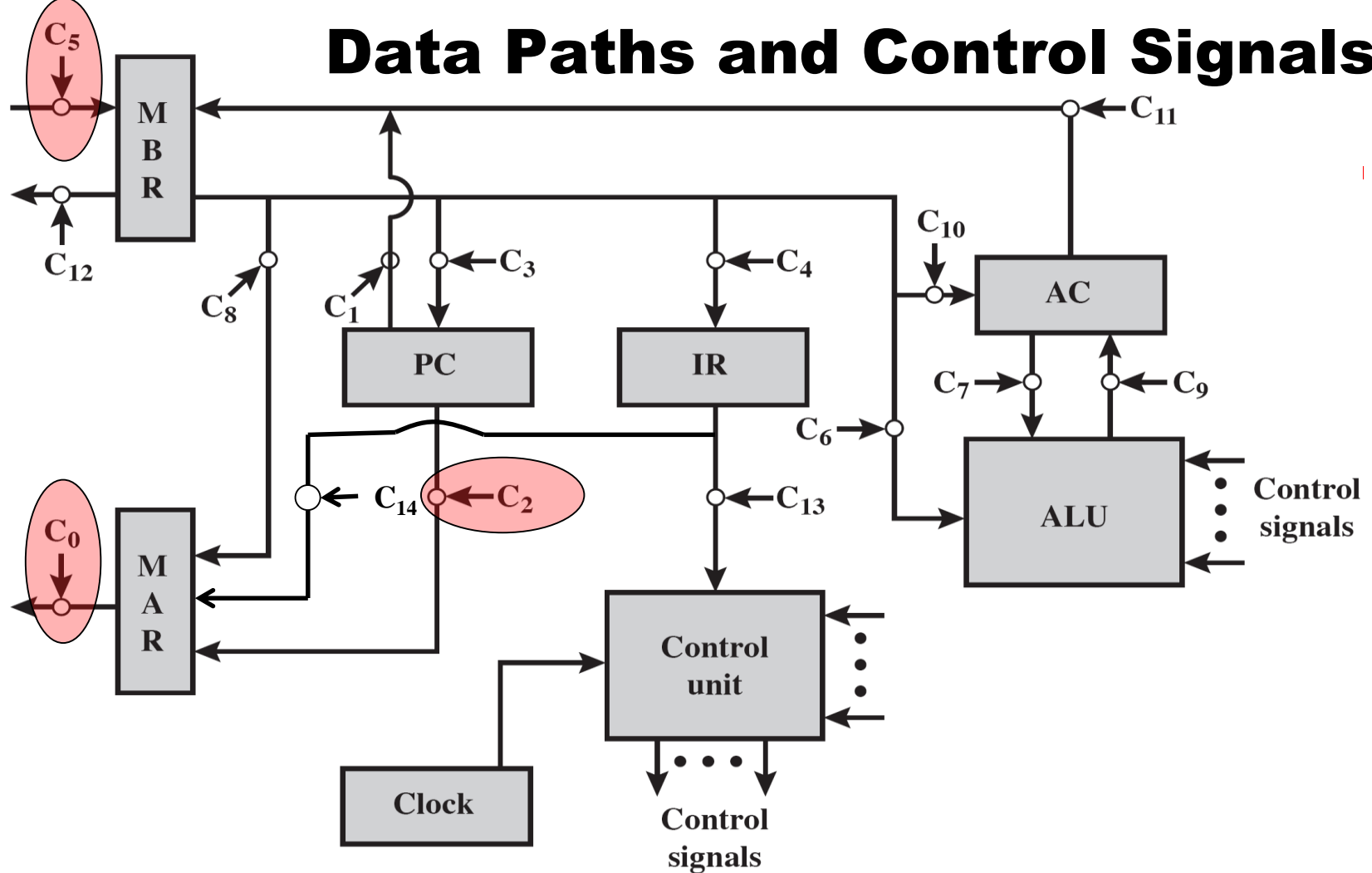
Model of the Control Unit



Control Unit – Inputs and Outputs

- **Inputs**
 - **Clock:**
 - One micro-operation (or set of parallel micro-operations) per clock cycle.
 - **Instruction register**
 - Opcode for current instruction.
 - Determines which micro-operations are performed.
 - **Flags**
 - State of CPU.
 - Results of previous operations.
 - **Control signals from control bus**
 - Interrupts.
 - Acknowledgements.
- **Outputs (control signals)**
 - **Within CPU**
 - Causes data movement.
 - Activates specific functions.
 - **Via control bus:** to memory and to I/O modules.

Data Paths and Control Signals



• MAR \leftarrow [PC] **Instruction Fetch**

- Control unit activates signal to open gates between PC and MAR: C_2
- MAR \leftarrow Memory
 - Open gates between MAR and address bus : C_0
 - Memory read control signal.: C_R
 - Open gates between data bus and MBR : C_5

Micro-operations		Timing	Active Control Signals
Fetch:	$t_1: \text{MAR} \leftarrow [\text{PC}]$		C_2
	$t_2: \text{MBR} \leftarrow \text{Memory}$ $\text{PC} \leftarrow [\text{PC}] + 1$		C_0, C_R, C_5
	$t_3: \text{IR} \leftarrow [\text{MBR}]$		C_4
Indirect:	$t_1: \text{MAR} \leftarrow [\text{IR-address}]$		C_{14}
	$t_2: \text{MBR} \leftarrow \text{Memory}$		C_0, C_R, C_5
	$t_3: \text{IR-address} \leftarrow [\text{MBR}]$		C_4
Interrupt:	$t_1: \text{MBR} \leftarrow [\text{PC}]$		C_1
	$t_2: \text{MAR} \leftarrow \text{Save-address}$ $\text{PC} \leftarrow \text{Routine-address}$		
	$t_3: \text{Memory} \leftarrow [\text{MBR}]$		C_0, C_W, C_{12}

C_R : Read control signal to system bus.

C_W : Write control signal to system bus.

Internal 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.
- Example: ADD X

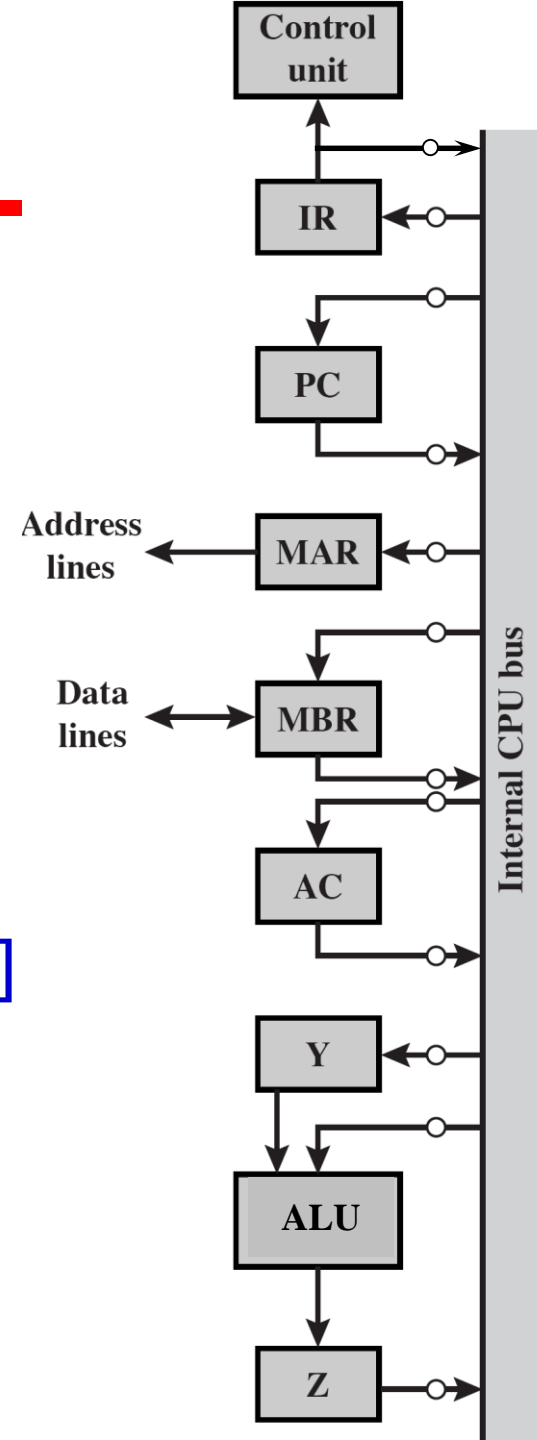
$t_1: \text{MAR} \leftarrow [\text{IR-address}]$

$t_2: \text{MBR} \leftarrow \text{Memory}$

$Y \leftarrow [\text{AC}]$

$t_3: Z \leftarrow [\text{MBR}] + [Y]$

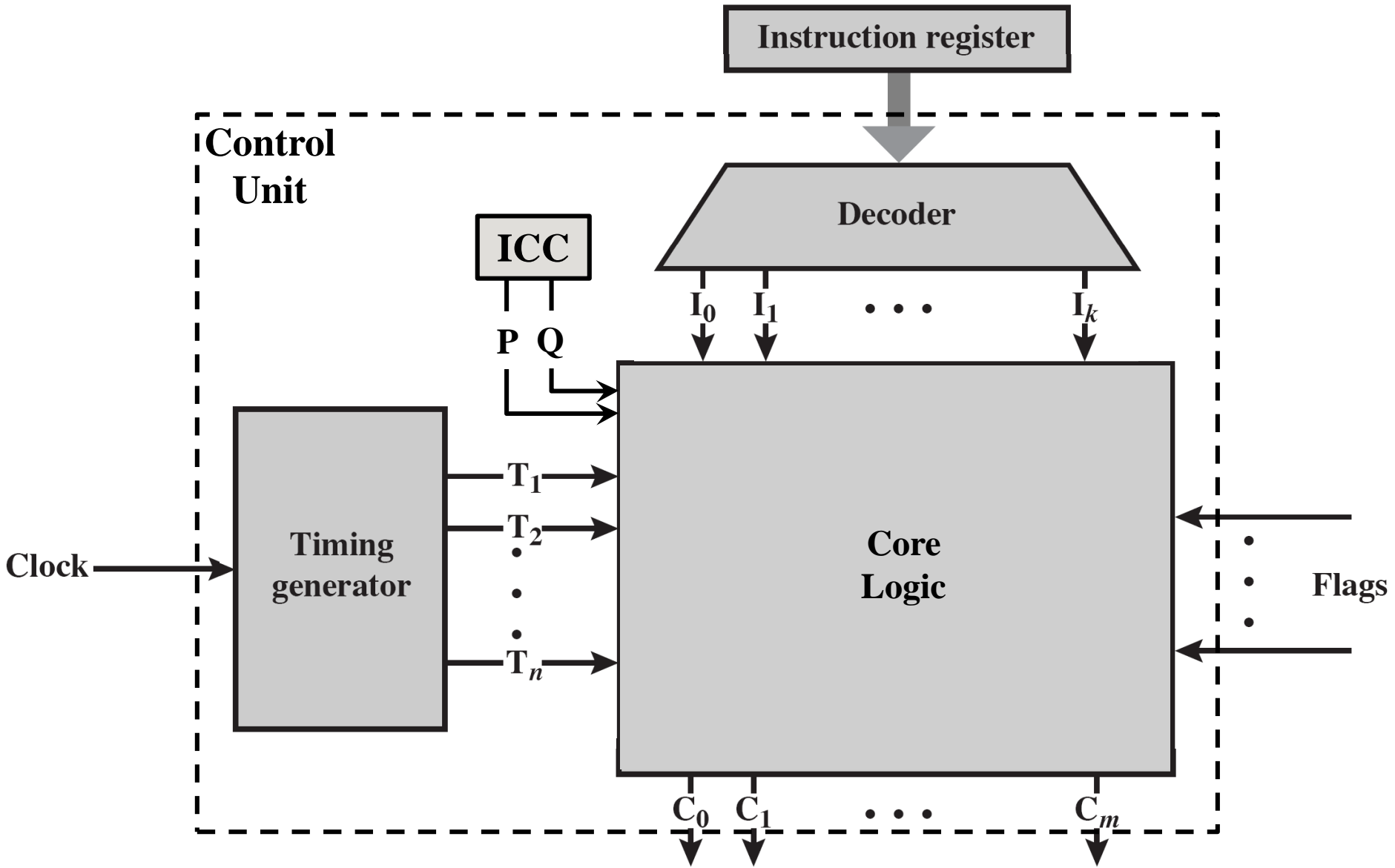
$t_4: \text{AC} \leftarrow [Z]$



Implementing the Control Unit

- A wide variety of techniques have been used.
- Most of them fall into one of two categories:
 1. **Hardwired implementation**
 - Control signals are generated by a state machine **circuit**.
 2. **Microprogrammed implementation**
 - Control signals are generated by a **program** similar to machine language programs.

Hardwired Implementation



Control Unit Core

- To design the core logic, we need to derive a Boolean expression for each control signal.
- Example: let's derive an expression for C_5 (read data from external bus to MBR)
 - Suppose instruction subcycles are encoded by control signals P & Q s.t. $PQ=00 \rightarrow$ Fetch, $PQ=01 \rightarrow$ Indirect, $PQ=10 \rightarrow$ Execute, $PQ=11 \rightarrow$ Interrupt.
 - Suppose C_5 should be **activated** in:
 - Step 2 (T_2) during fetch and indirect subcycles.
 - Step 3 (T_3) during execution of ADD (I_5) and AND (I_8) instructions.
 - Boolean expression for C_5 :

$$C_5 = \bar{P}.\bar{Q}.T_2 + \bar{P}.Q.T_2 + P.\bar{Q}.(I_5 + I_8).T_3$$

Problems with Hardwired Design

- Complex sequencing & micro-operation logic.
- Difficult to design and test.
- Inflexible design.
 - Difficult to add new instructions.

Reading Material

- Stallings, Chapter 19:
 - URL:
http://www.ecs.csun.edu/~cputnam/Comp546/Stallings-Appendices/19-Control_Unit.pdf
 - Pages 1 – 24
 - Pages 30 – 35