CAO: Lecture 23
Implementation of control unit, Enhancing performance with pipelining

Topics Covered

- Control unit
- Timing and control
- Timing signals
- Pipelining and vector processing
- Parallel processing
- Pipeline and multiple function units
- Instruction cycle

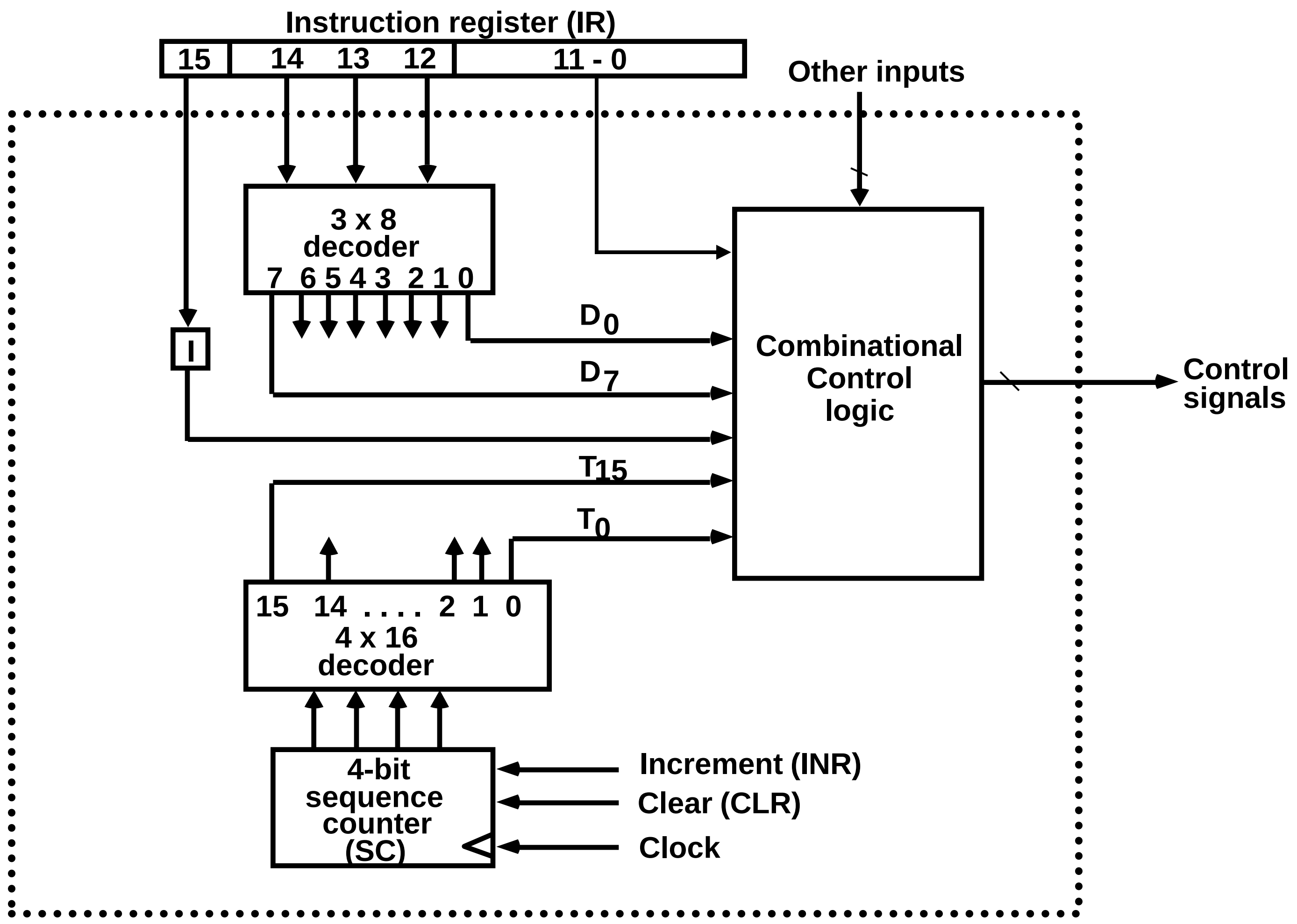
CONTROL UNIT

- Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them

- Control units are implemented in one of two ways
- *Hardwired* Control
  - CU is made up of sequential and combinational circuits to generate the control signals
- *Microprogrammed* Control
  - A control memory on the processor contains microprograms that activate the necessary control signals

- We will consider a hardwired implementation of the control unit for the Basic Computer

TIMING AND CONTROL

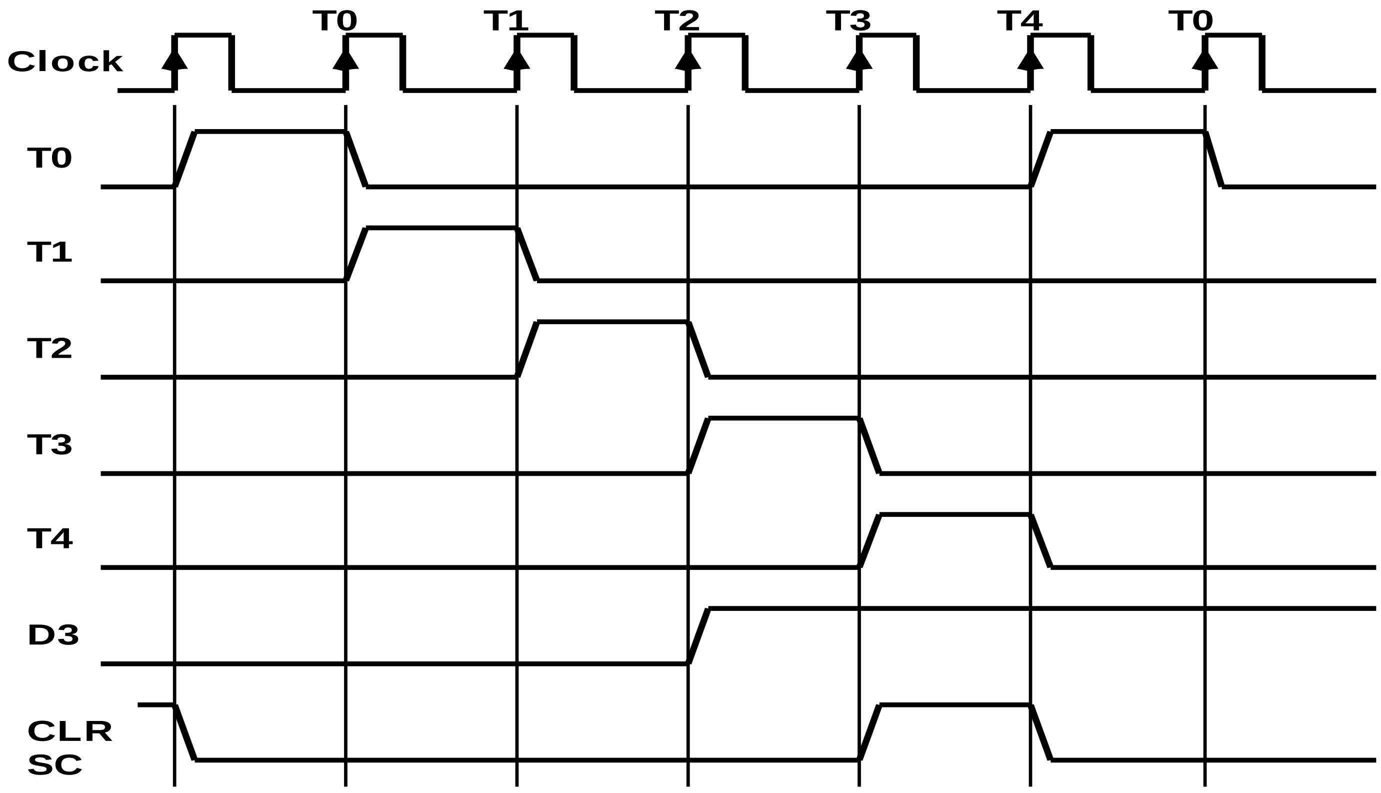


TIMING SIGNALS

- Example: T_0, T_1, T_2, T_3, T_4, T_0, T_1, . . .

Assume: At time T_4, SC is cleared to 0 if decoder output D3 is active.

$$D_3T_4: SC \leftarrow 0$$



PIPELINING AND VECTOR PROCESSING

- Parallel Processing
- Pipelining
- Arithmetic Pipeline
- Instruction Pipeline

PARALLEL PROCESSING

Execution of *Concurrent Events* in the computing process to achieve faster *Computational Speed* and throughput.

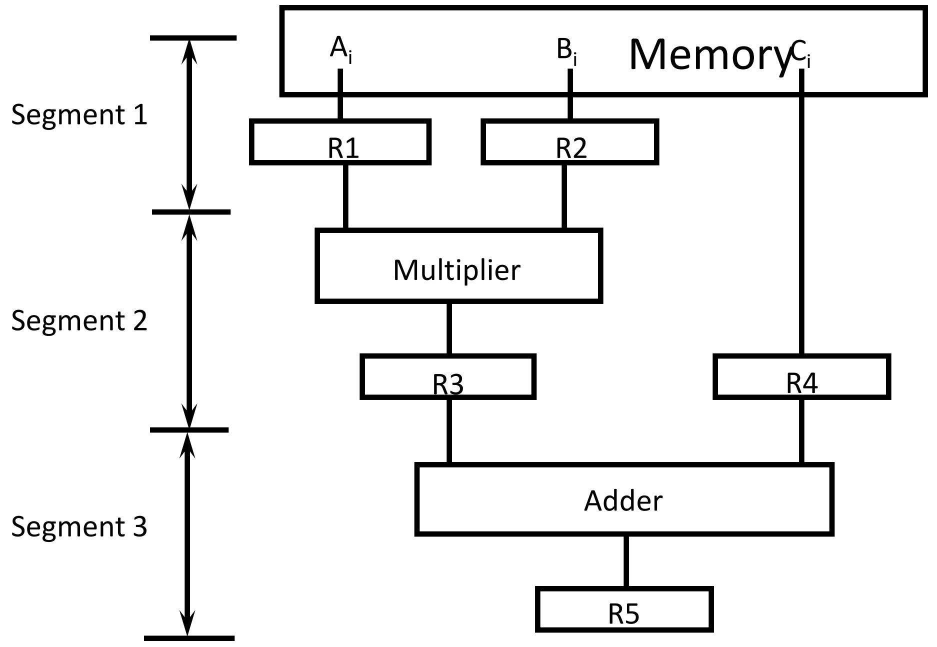
Levels of Parallel Processing

- Job or Program level
- Task or Procedure level
- Inter-Instruction level
- Intra-Instruction level

PIPELINING

A technique of decomposing a sequential process into suboperations, with each subprocess being executed in a partial dedicated segment that operates concurrently with all other segments.

$A_i * B_i + C_i$ for $i = 1, 2, 3, \ldots, 7$



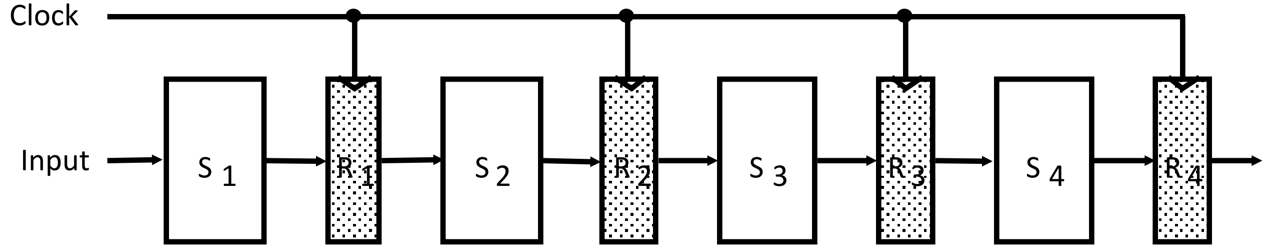
$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Load A_i and B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply and load C_i
$R5 \leftarrow R3 + R4$	Add

OPERATIONS IN EACH PIPELINE STAGE

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A1	B1			
2	A2	B2	A1 * B1	C1	
3	A3	B3	A2 * B2	C2	A1 * B1 + C1
4	A4	B4	A3 * B3	C3	A2 * B2 + C2
5	A5	B5	A4 * B4	C4	A3 * B3 + C3
6	A6	B6	A5 * B5	C5	A4 * B4 + C4
7	A7	B7	A6 * B6	C6	A5 * B5 + C5
8			A7 * B7	C7	A6 * B6 + C6
9					A7 * B7 + C7

GENERAL PIPELINE

General Structure of a 4-Segment Pipeline



Space-Time Diagram

Segment	1	2	3	4	5	6	7	8	9
1	T1	T2	T3	T4	T5	T6			
2		T1	T2	T3	T4	T5	T6		
3			T1	T2	T3	T4	T5	T6	
4				T1	T2	T3	T4	T5	T6

Clock cycles

PIPELINE SPEEDUP

n: Number of tasks to be performed

Conventional Machine (Non-Pipelined)

t_n: Clock cycle

τ_1: Time required to complete the n tasks

$\tau_1 = n * t_n$

Pipelined Machine (k stages)

t_p: Clock cycle (time to complete each suboperation)

τ_κ: Time required to complete the n tasks

$\tau_\kappa = (k + n - 1) * t_p$

Speedup

S_k: Speedup

$S_k = n*t_n / (k + n - 1)*t_p$

$$\lim_{n \to \infty} S_k = \frac{t_n}{t_p} \quad (= k, \text{ if } t_n = k * t_p)$$

PIPELINE AND MULTIPLE FUNCTION UNITS

Example

- 4-stage pipeline
- subopertion in each stage; t_p = 20nS
- 100 tasks to be executed
- 1 task in non-pipelined system; 20*4 = 80nS

Pipelined System

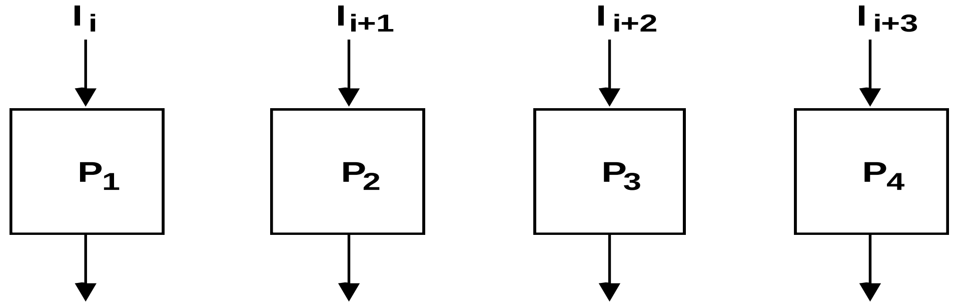
$(k + n - 1)*t_p = (4 + 99) * 20 = 2060$nS

Non-Pipelined System

$n*k*t_p = 100 * 80 = 8000$nS

Speedup

$S_k = 8000 / 2060 = 3.88$



4-Stage Pipeline is basically identical to the system with 4 identical function units

Multiple Functional Units

ARITHMETIC PIPELINE

Floating-point adder

$X = A \times 2^a$

$Y = B \times 2^b$

[1] Compare the exponents

[2] Align the mantissa

[3] Add/sub the mantissa

[4] Normalize the result

X=0.9504* 103

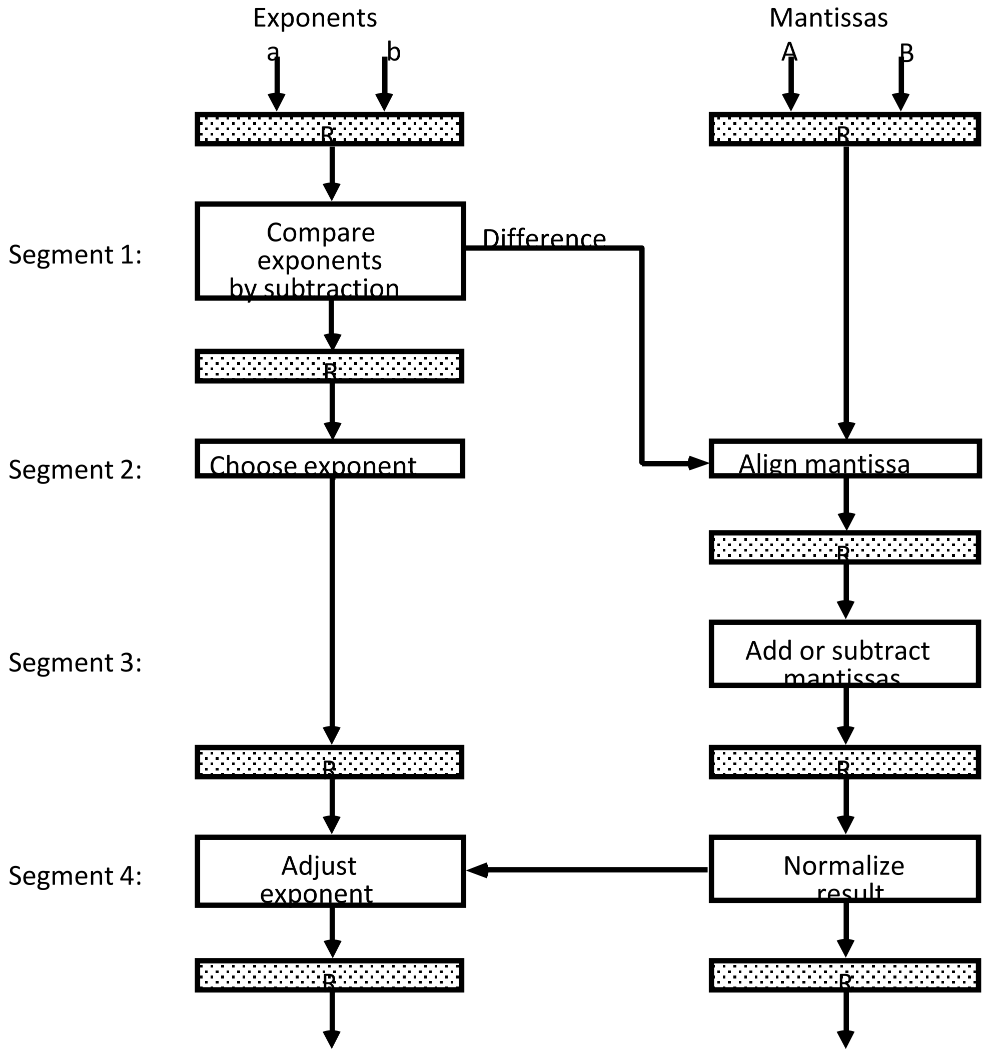
Y= 0.8200*102

X=0.9504* 103

Y= 0.0820*103

Z=1.0324*103

Z=0.1032*103



INSTRUCTION CYCLE

Six Phases* in an Instruction Cycle

[1] Fetch an instruction from memory
[2] Decode the instruction
[3] Calculate the effective address of the operand
[4] Fetch the operands from memory
[5] Execute the operation
[6] Store the result in the proper place

* Some instructions skip some phases
* Effective address calculation can be done in the part of the decoding phase
* Storage of the operation result into a register is done automatically in the execution phase

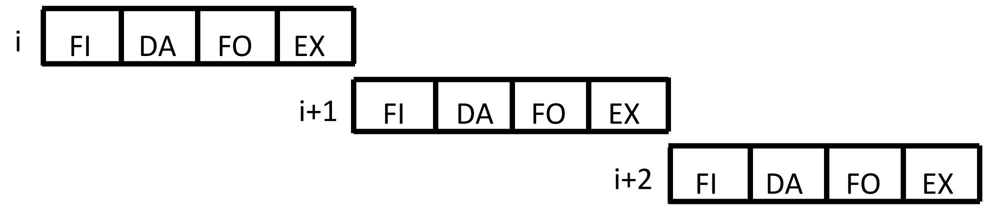
==> 4-Stage Pipeline

[1] FI: Fetch an instruction from memory
[2] DA: Decode the instruction and calculate the effective address of the operand
[3] FO: Fetch the operand
[4] EX: Execute the operation

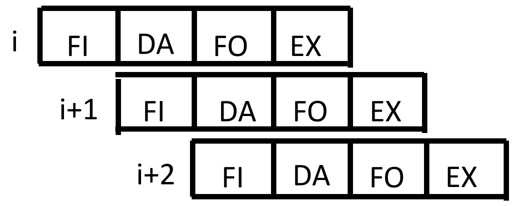
INSTRUCTION PIPELINE

Execution of Three Instructions in a 4-Stage Pipeline

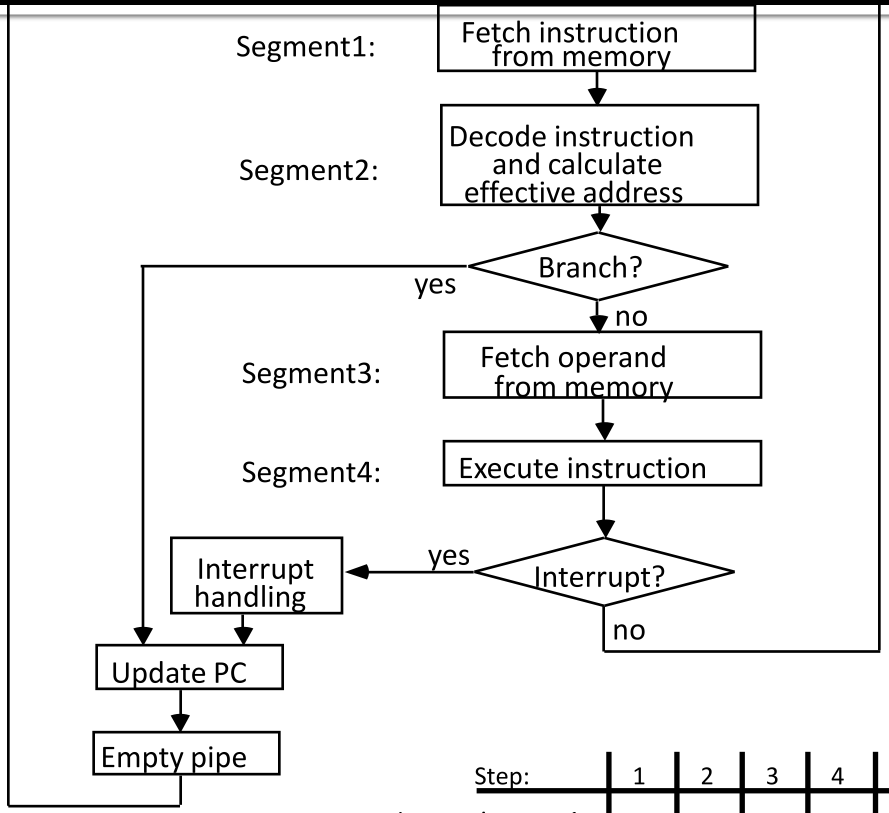
Conventional



Pipelined



INSTRUCTION EXECUTION IN A 4-STAGE PIPELINE



	Step:	1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction	1	FI	DA	FO	EX									
	2		FI	DA	FO	EX								
(Branch)	3			FI	DA	FO	EX							
	4				FI	-	-	FI	DA	FO	EX			
	5					-	-	-	FI	DA	FO	EX		
	6									FI	DA	FO	EX	
	7										FI	DA	FO	EX