PROCESSING UNIT DESIGN

Danang Wahyu Utomo
danang.wu@dsn.dinus.ac.id
+6285 725 158 327

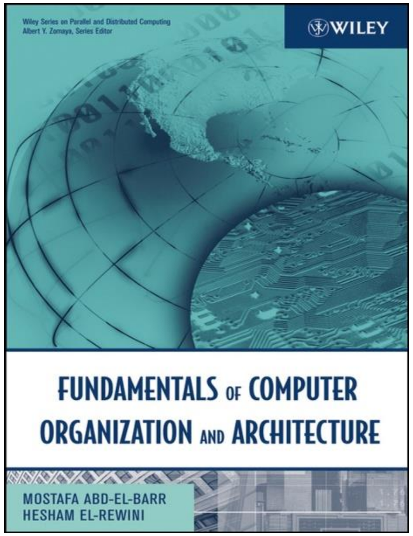
RENCANA KEGIATAN PERKULIAHAN SEMESTER

W	Pokok Bahasan
1	Organisasi dan Arsitektur Komputer
2	Sistem Komputer
3	Instruction Set Architecture and Design
4	
5	Computer Arithmetic
6	
7	Review Materi 1-6
8	**Ujian Tengah Semester**

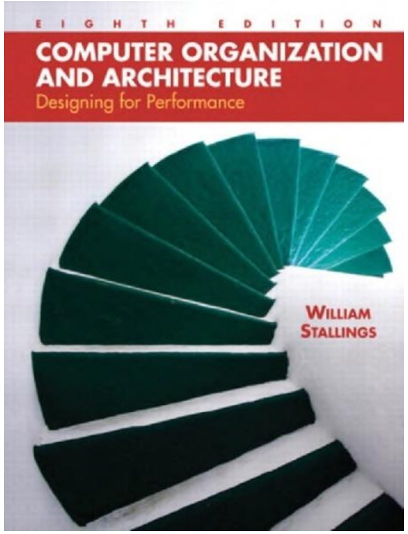
W	Pokok Bahasan
9	**Desain Unit Pemrosesan**
10	
11	Desain Sistem Memory
12	
13	Desain dan Organisasi Input Output
14	
15	Teknik Desain Pipelining
16	**Ujian Akhir Semester**

Reference

- Mustafa Abd-el-Bhar, Hesham El Rewini – Fundamentals of Computer Organization and Architecture 9th Edition (2004)


Wiley Series on Parallel and Distributed Computing
Albert Y. Zomaya, Series Editor
WILEY
FUNDAMENTALS OF COMPUTER ORGANIZATION AND ARCHITECTURE
MOSTAFA ABD-EL-BARR
HESHAM EL-REWINI

- William Stallings – Computer Organization and Architecture Designing For Performance 8th Edition (2009)

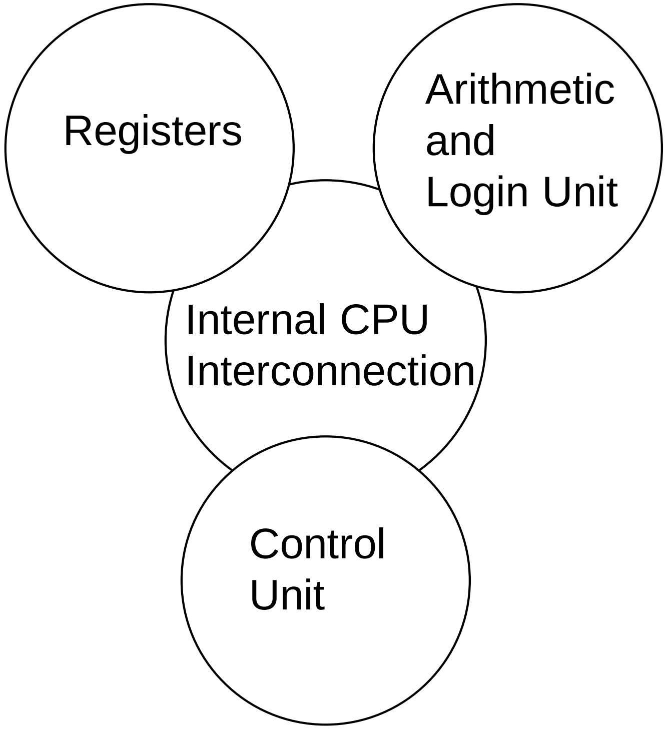

EIGHTH EDITION
COMPUTER ORGANIZATION AND ARCHITECTURE
Designing for Performance
WILLIAM STALLINGS

Content

- CPU Basics
- Register Set
- CPU Instruction Cycle
- Control Unit

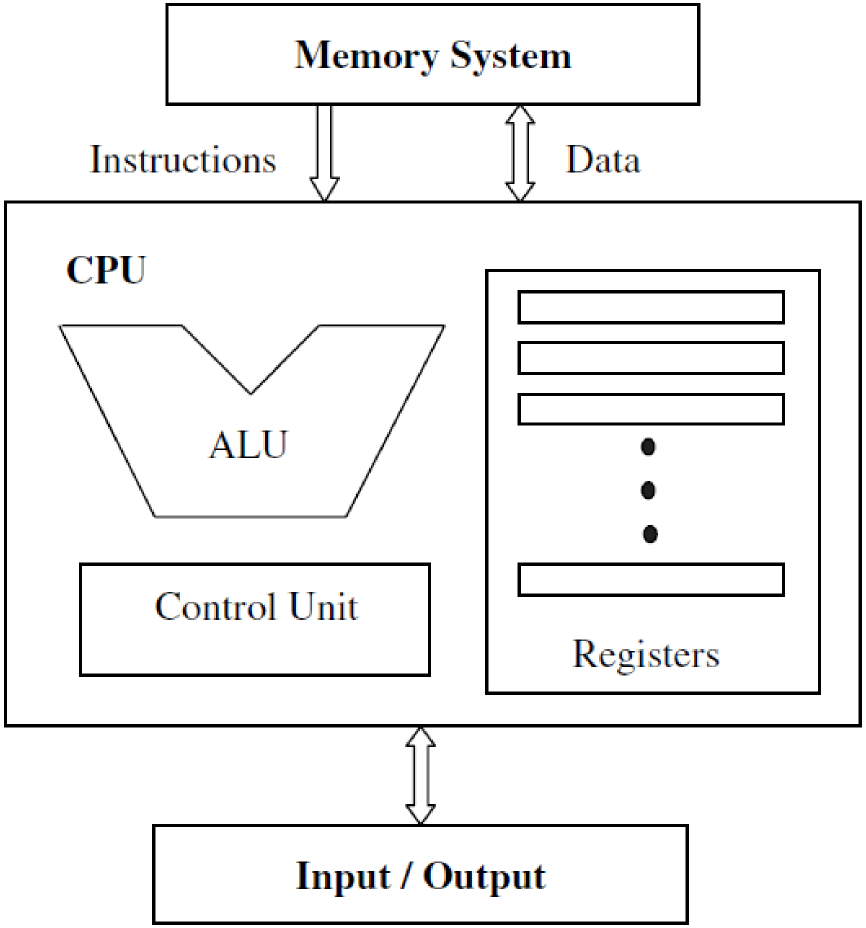
CPU Basics

- CPU has three major component :
  - Register
  - Arithmetic Logic Unit (ALU)
  - Control Unit



CPU Basics

- CPU fetch instruction from memory
- Reads and writes data from and to memory
- Transfer data from and to I/O device



CPU Basics

- Execution cycle :
  - The next instruction to be executed, whose address is obtained from the PC, is fetches from the memory and stored in the IR
  - The instruction is decoded
  - Operand are fetched from the memory and stored in CPU register, if needed
  - Instruction is needed
  - Result are transferred from CPU register to the memory, if needed

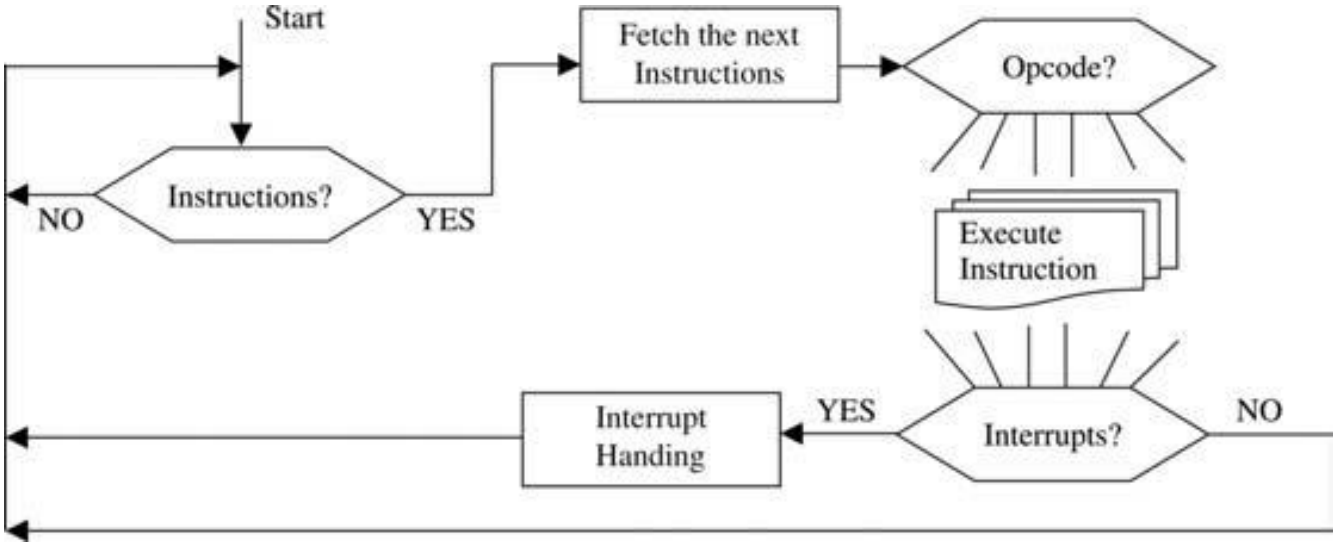
Execution Cycle

- The execution cycle is repeated as long as there are more instruction to execute
- Interrupts is usually included in the execution cycle.

  Example : I/O device request, arithmetic overflow, page fault
- When an interrupt request is encountered, a transfer to an interrupt handling routine
- Interrupt handling routine are program that invoked to collect:
  - The state of the currently executing program
  - Correct the cause of the interrupt
  - Restore the state of the program

Instruction Cycle

- When instruction to be executed, the next instruction is fetched from main memory
- The instruction based on the operation specified in the opcode field of the instruction
- When interrupt, an interrupt handling needs to be invoked



Register Set

- Register are essentially memory locations within the CPU that are used to create and store the result of CPU operations and calculations.
- General-purpose register used for multiple purposes and assigned to a variety of functions by the programmer

  Ex : floating-point and stack operation, addressing functions
- Special-purpose register are restricted to only specified function

  Ex : some register used to hold data and cannot be used in the calculations of operand addresses.

Register Set

- Two register are essential in memory write and read operations
  - Memory Data Register

    the word to be stored into the memory location is first loaded by the CPU into a specified register
  - Memory Address Register

    the address of the location into which the word is to be stored is loaded by the CPU into a specified register

MDR & MAR

- In write operation, MDR and MAR are used as follows :
  - The word to be stored into the memory location is first loaded by the CPU into MDR
  - The address of the location which the word is to be stored is loaded by the CPU into MAR
  - A write signal is issued by the CPU

MDR & MAR

- In memory read operation, MDR and MAR are used as follows :
  - The address of the location from which the word is to be read is loaded into the MAR
  - A read signal is issued by the CPU
  - The required word will be loaded by the memory into the MDR ready for use by the CPU

Assembly

- A computer can be execute program only when they are represented in machine language specific to its architecture
- An assembly language program is a symbolic representation of the machine language program
- Machine language is pure binary code, whereas assembly language is a direct mapping of the binary code onto a symbolic form

Simple Machine

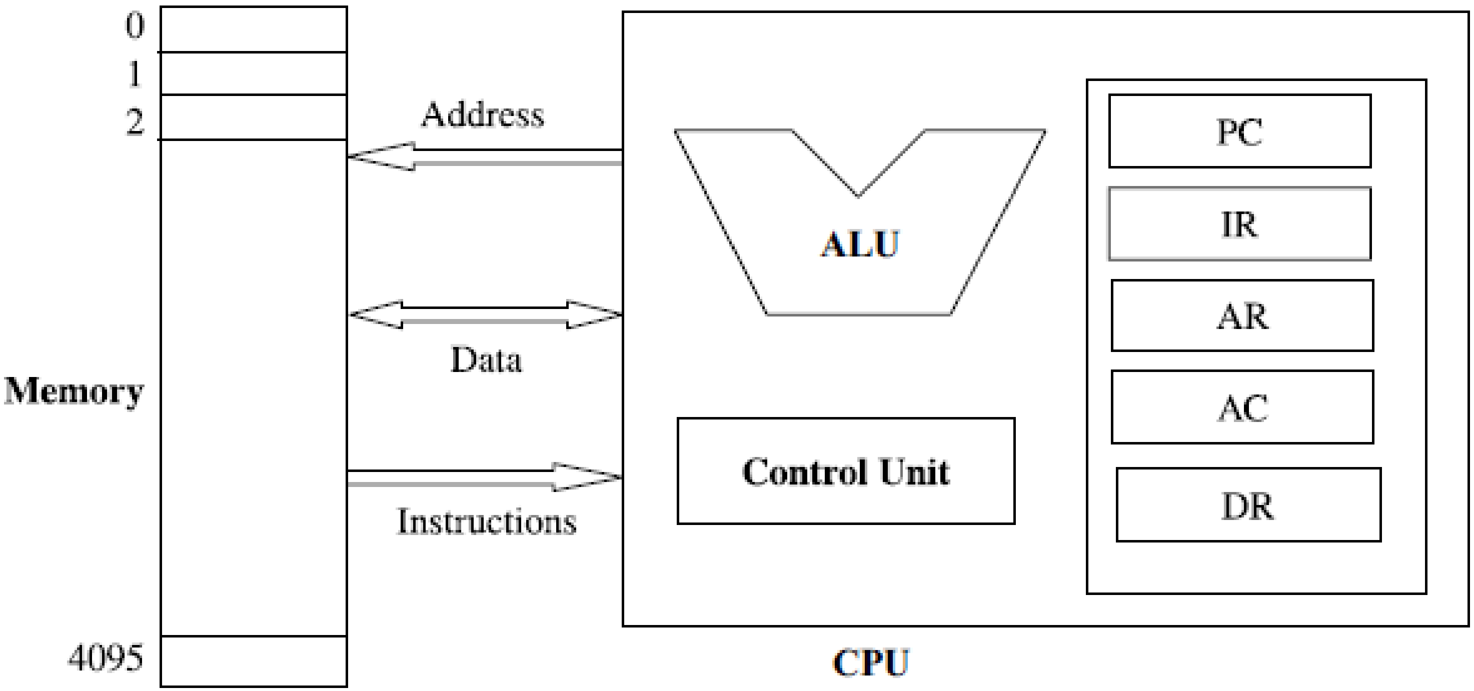
- Machine language is the native language of a given processor
- Each different type of processor has its own unique assembly language
- Simple machine is an accumulator based processor which have five 16-bit registers (hypothetical machine) :
  - PC – Program Counter
  - IR – Instruction Register
  - AR – Address Register
  - AC – Accumulator
  - DR – Data Register

Simple Machine

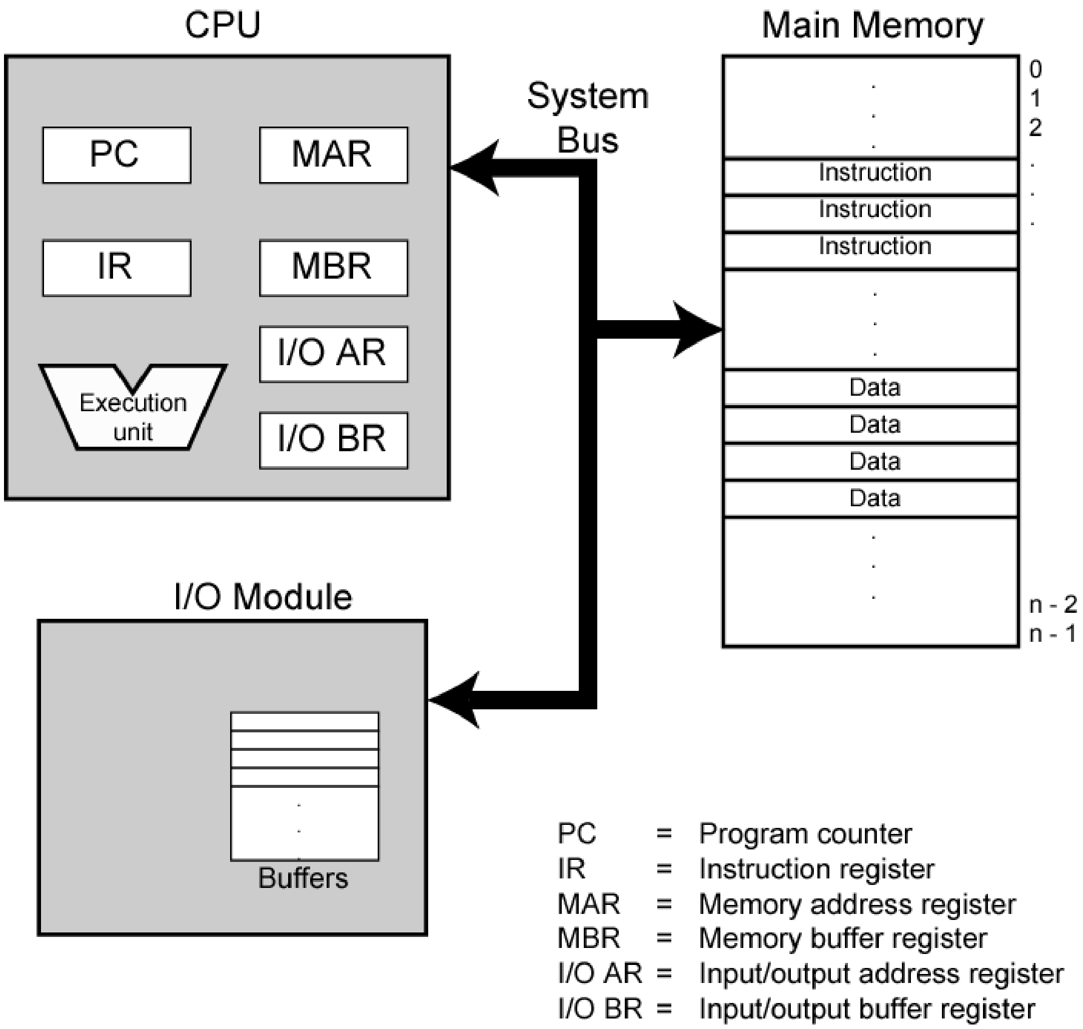
- Program Counter
  - Contains **the address** of the next instruction to be executed
- Instruction Register
  - Contains the **operation code** portion of the **instruction** being executed
- Address Register
  - Contains the address portion of the instruction being executed
- Accumulator
  - Destination of **data**
- Data Register
  - Hold data

The word size 16 bits

Simple Machine



Simple Machine



PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

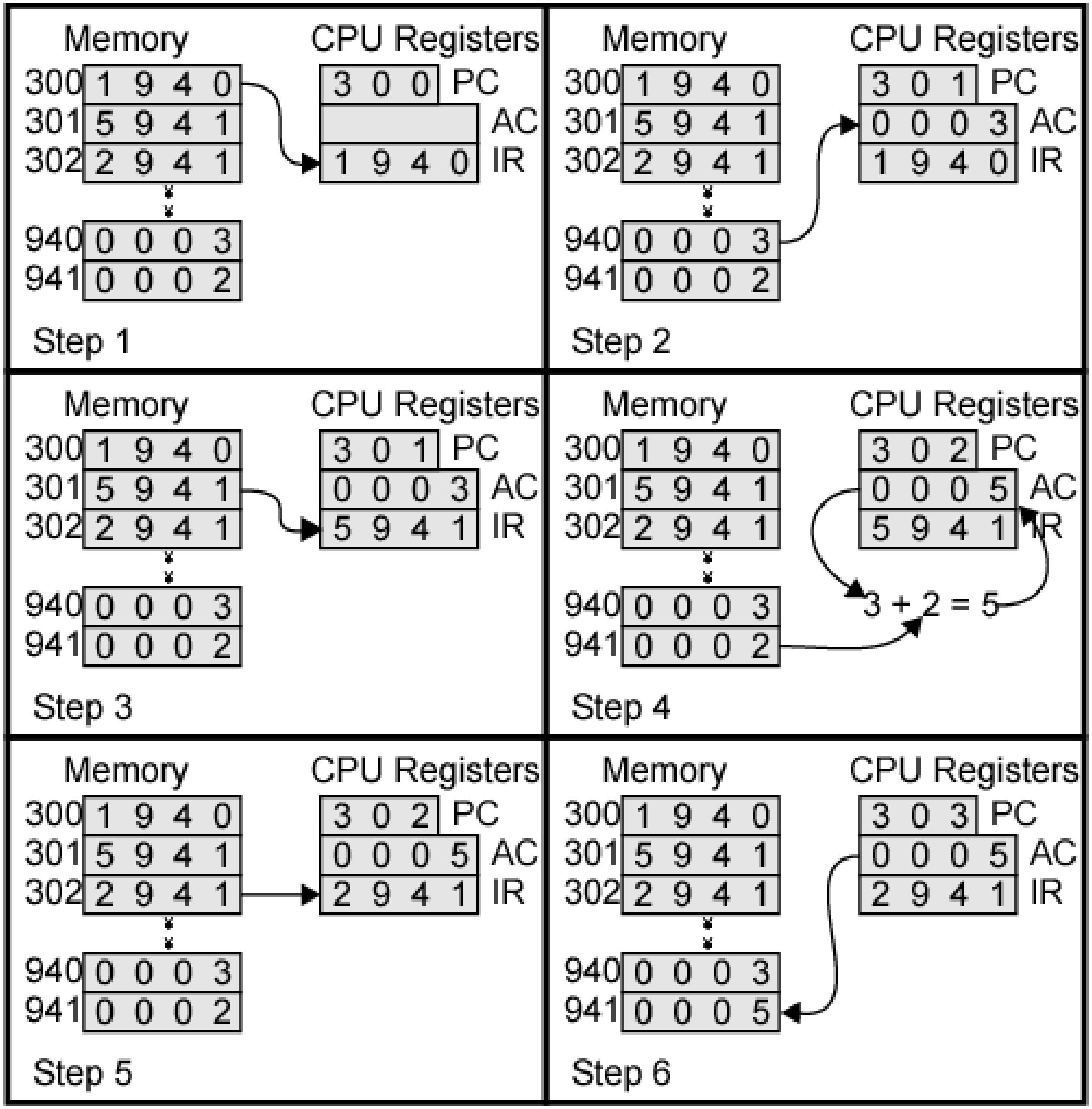
Simple Machine

- Simple processor supports three types of instructions :
  - Data transfer

    load, store, and move data between the registers AC and DR
  - Data processing

    add, subtract, and, not
  - Program control

    jump and conditional jump
- The instruction size is 16 bits :
  - 4 bits for the **operation code (opcode)**
  - 12 bits for **the address**



Memory

300 1 9 4 0
301 5 9 4 1
302 2 9 4 1

940 0 0 0 3
941 0 0 0 2

CPU Registers

3 0 0 PC
AC
1 9 4 0 IR

Step 1

Memory

300 1 9 4 0
301 5 9 4 1
302 2 9 4 1

940 0 0 0 3
941 0 0 0 2

CPU Registers

3 0 1 PC
0 0 0 3 AC
1 9 4 0 IR

Step 2

Memory

300 1 9 4 0
301 5 9 4 1
302 2 9 4 1

940 0 0 0 3
941 0 0 0 2

CPU Registers

3 0 1 PC
0 0 0 3 AC
5 9 4 1 IR

Step 3

Memory

300 1 9 4 0
301 5 9 4 1
302 2 9 4 1

940 0 0 0 3
941 0 0 0 2

CPU Registers

3 0 2 PC
0 0 0 5 AC
5 9 4 1 IR

3 + 2 = 5

Step 4

Memory

300 1 9 4 0
301 5 9 4 1
302 2 9 4 1

940 0 0 0 3
941 0 0 0 2

CPU Registers

3 0 2 PC
0 0 0 5 AC
2 9 4 1 IR

Step 5

Memory

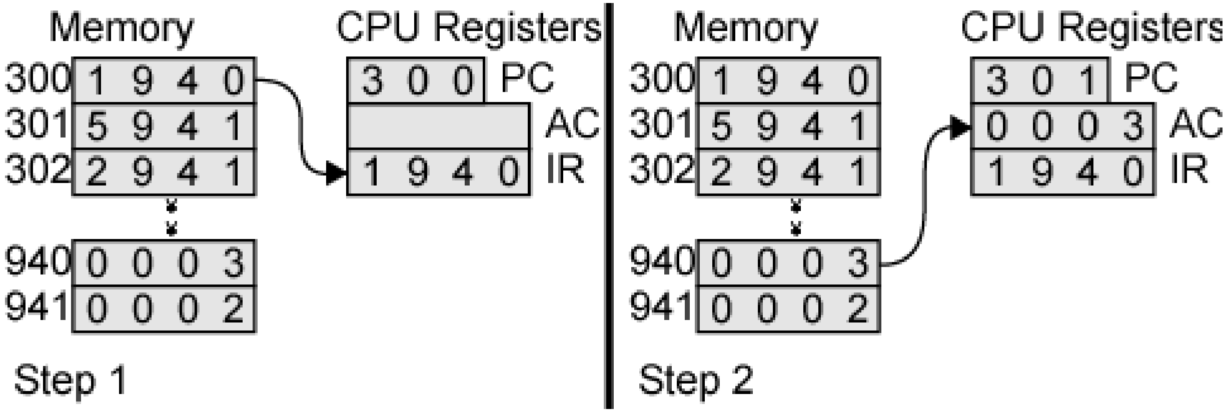
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1

940 0 0 0 3
941 0 0 0 5

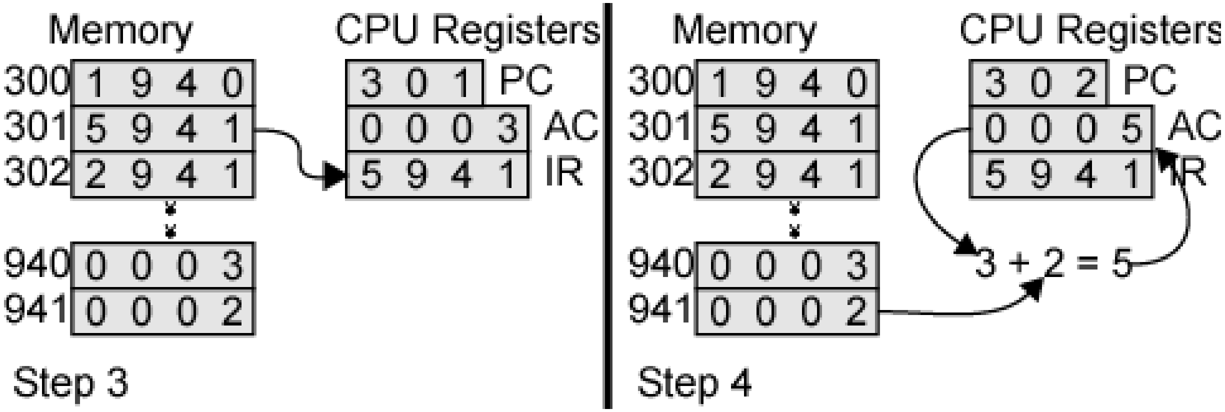
CPU Registers

3 0 3 PC
0 0 0 5 AC
2 9 4 1 IR

Step 6

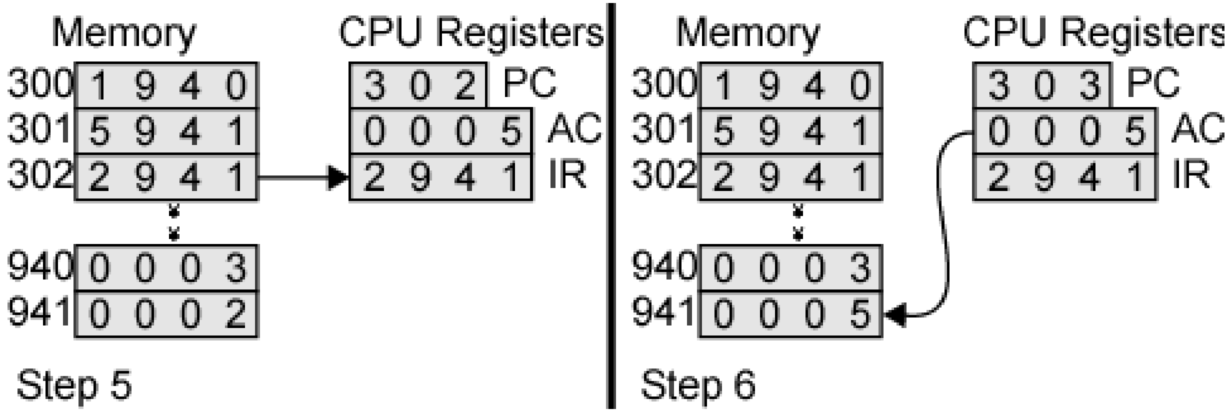


1. The PC contain 300, the address of the first instruction. This instruction (value 1940) is loaded into the instruction register IR and the PC is incremented
2. The first 4 bits (first hexadecimal digit) in the IR indicate that the AC is to be loaded. The remaining 12 bits (three hexadecimal digit) specify the address (940) from which data are to be loaded



3. The next instruction (5941) is fetched from location 301 and the PC is incremented

4. The old contents of the AC and the contents of location 941 are added and the result is stored in the AC



5. The next instruction (2941) is fetched from location 302, and the PC is incremented

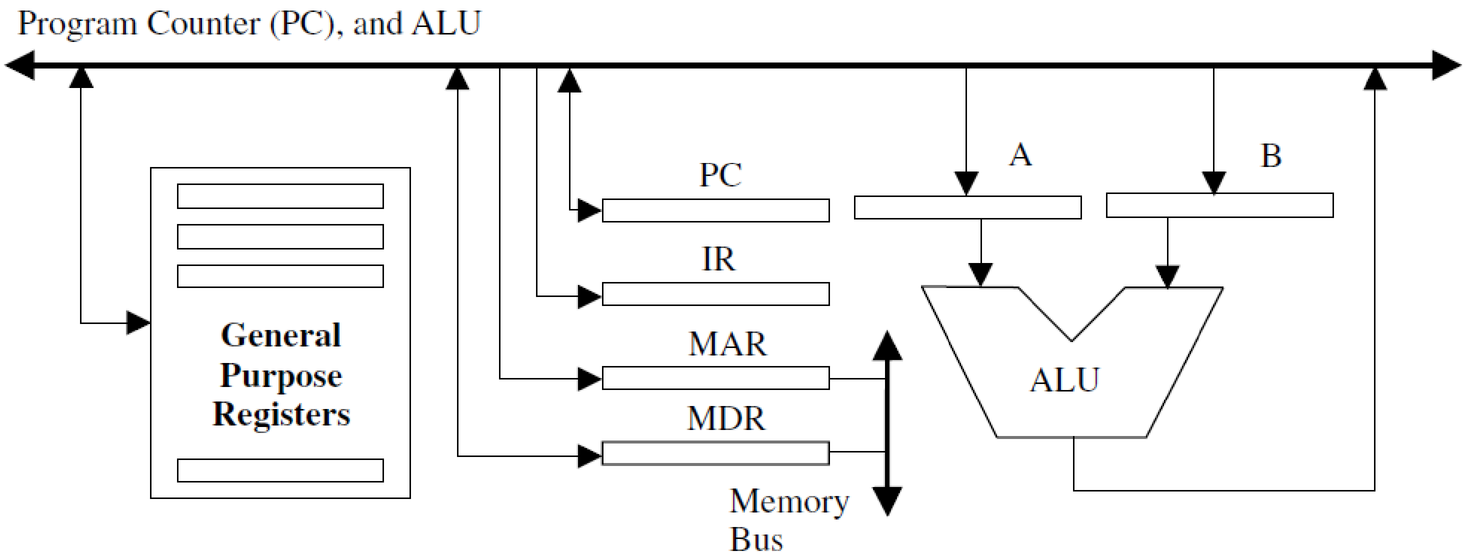
6. The content of the AC are stored in location 941

Datapath

- The CPU can be divided into a data section and control section
- The data section is also called the datapath (contains the register and the ALU)
- The datapath is capable of performing certain operations on data items
- Internal data movement among register and between the ALU and the Register may be carried out using different organization (**one bus, two bus, three bus**)

One-Bus Organization

- CPU register and ALU use a single bus to move outgoing and incoming data
- Only single data movement within one clock cycle
- Limit transfer in the same clock cycle (slow down the performance)

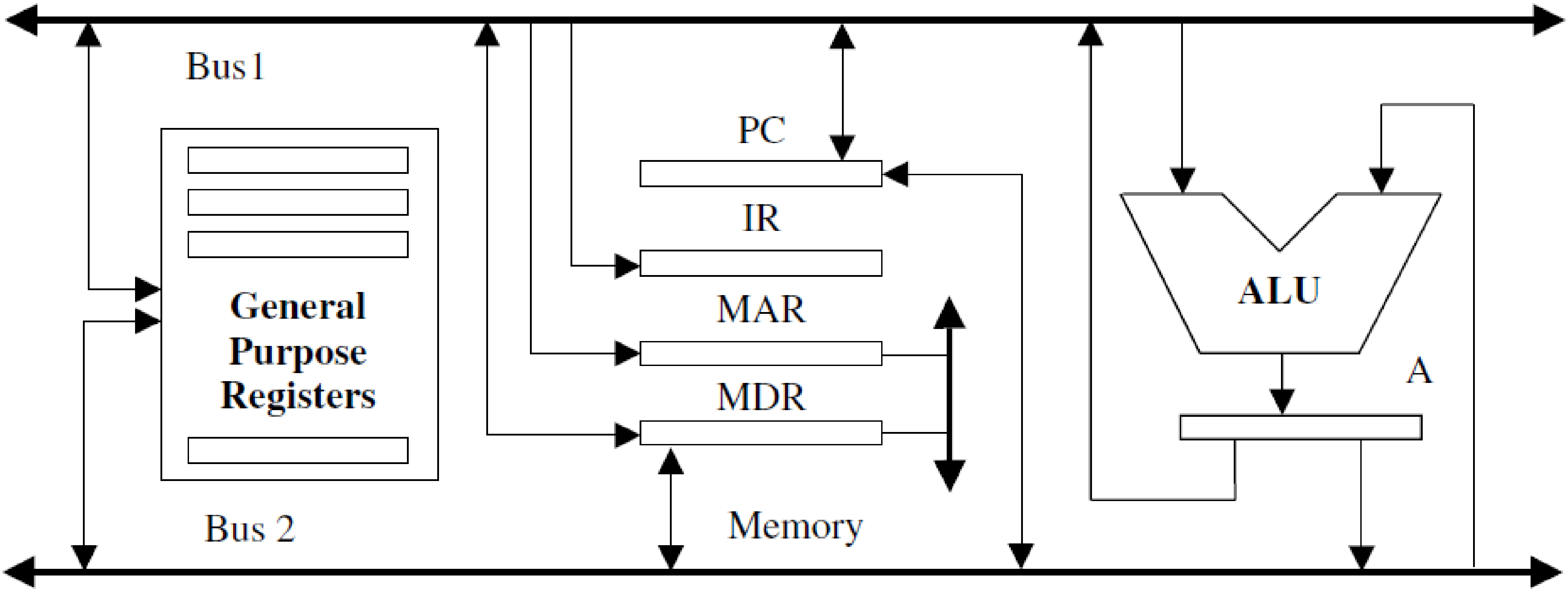


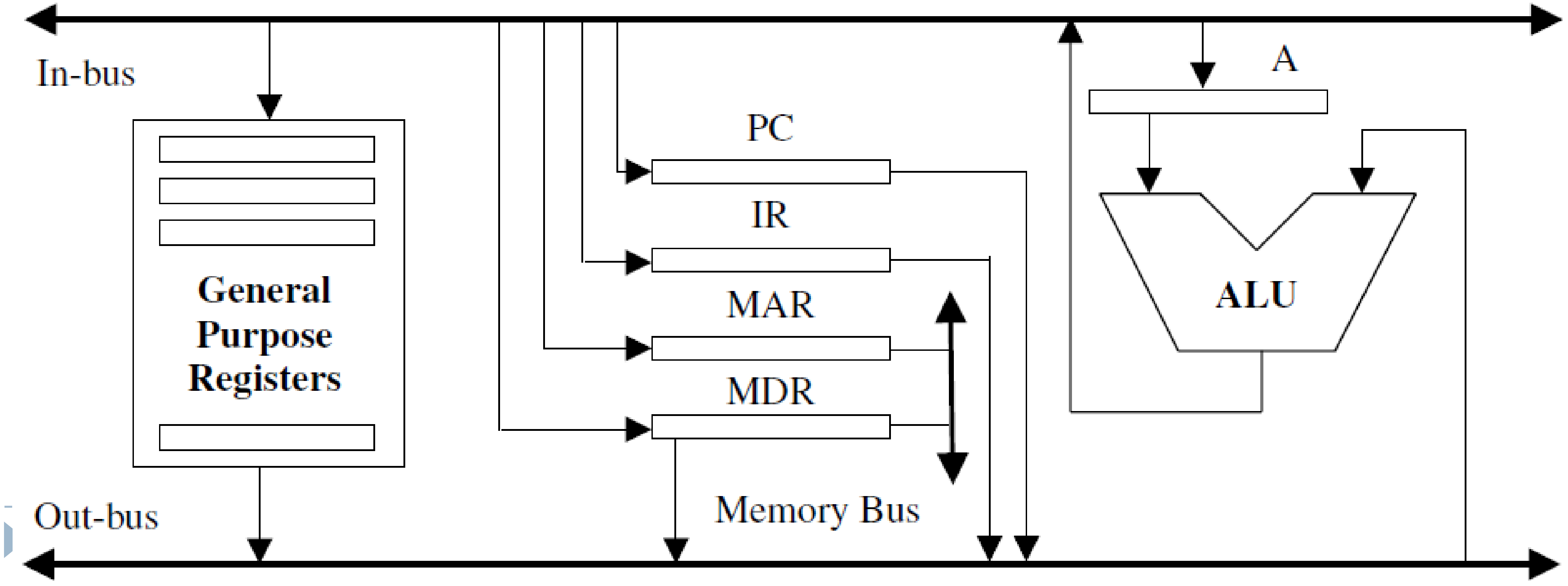
Two-Bus Organization

- General-purpose register are connected to both buses
- Data can be transferred from two different registers to the ALU at the same time
- Two operand operation can fetch both operands in the same clock cycle
- An additional buffer register may be needed to hold the output of the ALU when the two buses are busy

  Example :

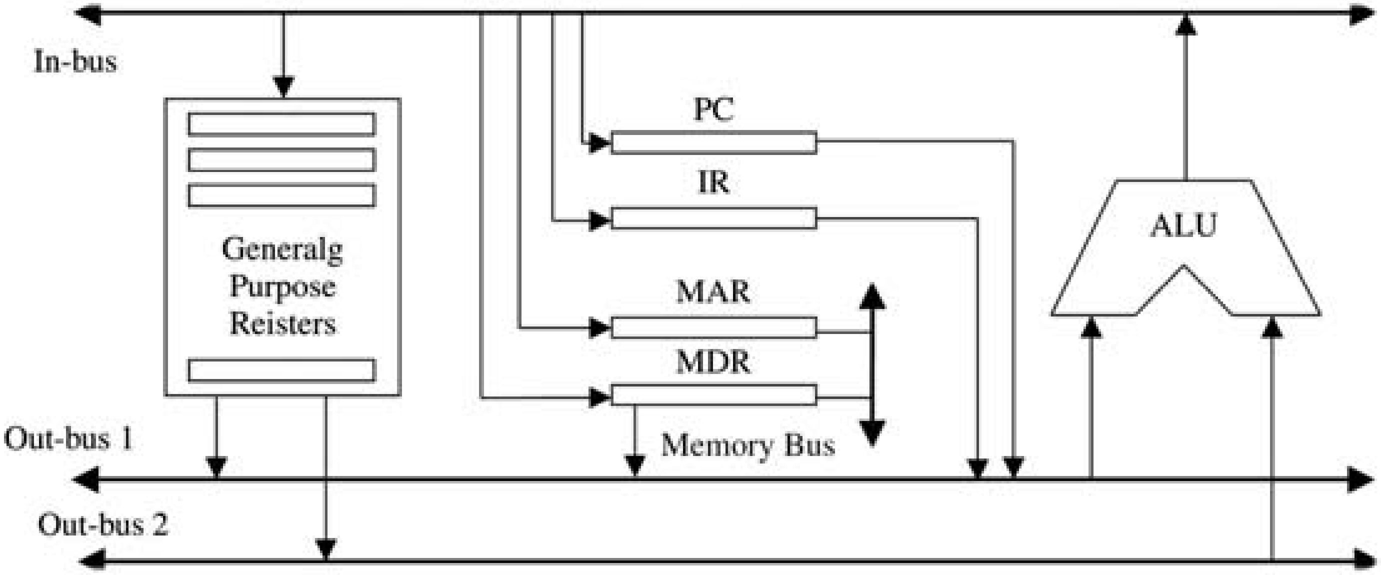
  one bus for moving data into register, another bus for transferring data out of the register


Bus1
PC
IR
MAR
MDR
General Purpose Registers
ALU
A
Memory
Bus 2


In-bus
A
PC
IR
MAR
MDR
General Purpose Registers
ALU
Memory Bus
Out-bus

Three-Bus Organization

- Two buses used as source buses while the third is used as destination.
- The source buses move data out of register (out-bus)
- The destination bus move data into a register (in-bus)



Fetch Instruction

- The contents of the PC are loaded into the MAR
- The value in the PC is incremented
- The instruction is loaded into the MDR
- The contents of the MDR are loaded into the IR

Step	Micro-Operation
T0	MAR ← (PC) ; A ← (PC)
T1	MDR ← Mem[MAR]; PC ← (A) + 4
T2	IR ← (MDR)

Interrupt Handling

- Check pending interrupt, if there is an interrupt request waiting. The steps are :
  - The contents of PC are loaded into MDR (saved)
  - The MAR is loaded with the address at which the PC contents are to be saved
  - The PC is loaded with the address of the first instruction of the interrupt handling routine
  - The contents of MDR (old value of the PC) are stored in memory

Step	Micro-Operation
T0	MDR ← (PC)
T1	MAR ← address 1 (save old PC)
	PC ← address 2 (interrupt handling routine)
T2	Mem[MAR] ← (MDR)

TERIMA KASIH