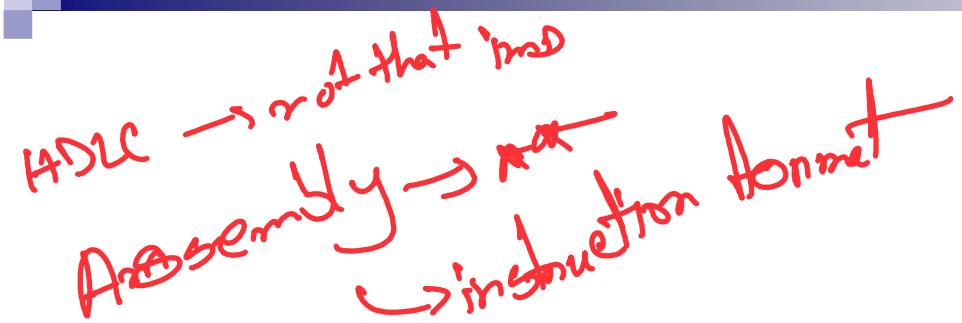


Lecture - 04

CPU Organization

- The primary function of the CPU is to execute sequences of instructions, which are stored in memory.

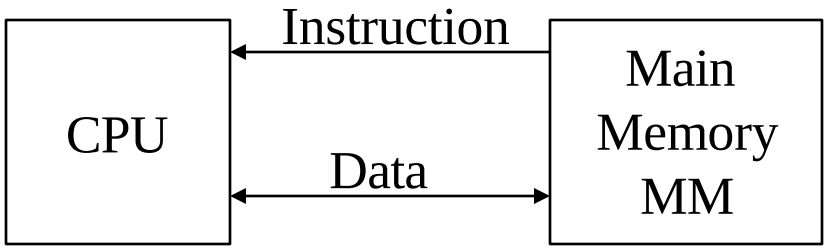
- The execution is carried out in three steps:

1. The CPU transfer instruction and when necessary, their input data (operand) from main memory to registers in the CPU.

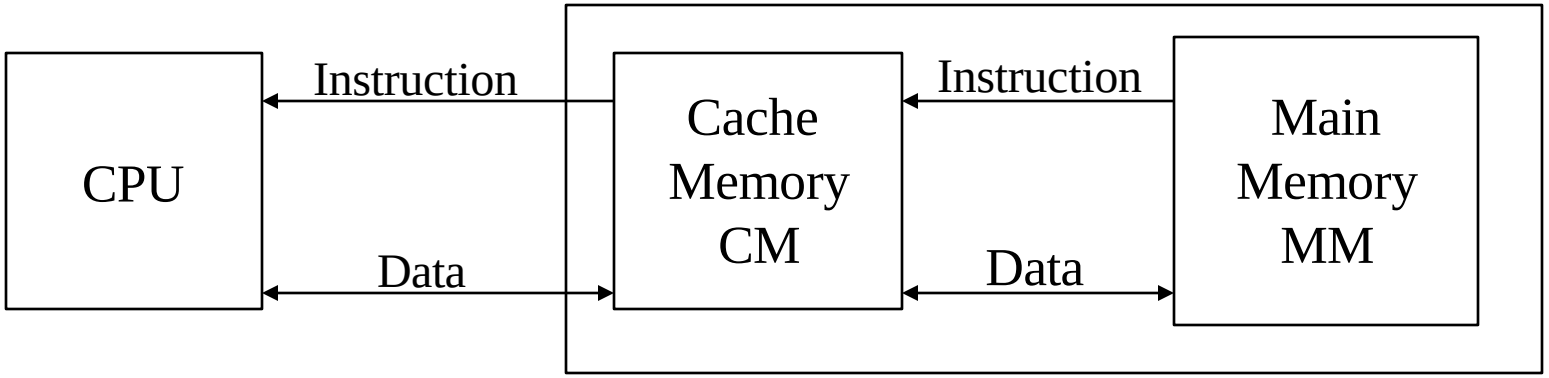
2. The CPU executes the instruction in their stored sequence except when the execution sequence is explicitly altered by a branch instruction.

3. When necessary, the CPU transfers data (results) from the CPU registers to main memory.

External Communication



Processor-memory communication without a cache



Processor-memory communication with a cache

External Communication

- CPU can perform a memory load or store operation from cache in a single clock cycle. On the other hand, the same operations take many clock cycles if they are performed from main memory.

- CPU considers the cache and the main memory as a single, seamless memory space consisting of 2^m addressable storage locations M(0), M(1), …. M(2^m-1).

Communication with IO Devices

- CPU communicates with IO devices in much the same way as it communicates with external memory.
- The IO devices are associated with addressable registers called ***IO ports*** to which the CPU can store a word or from which it can load a word.

Communication with IO Devices

- In some computers, memory locations and IO ports share the same set of addresses, so an address bit pattern that is assigned to memory cannot also be assigned to an IO port, and vice versa. This approach is called **memory–mapped IO.**

- Some computers employ IO instructions that are distinct from memory-referencing instructions. These instructions produce control signals to which IO ports, but not memory locations, respond. This approach is called **IO-mapped IO.**

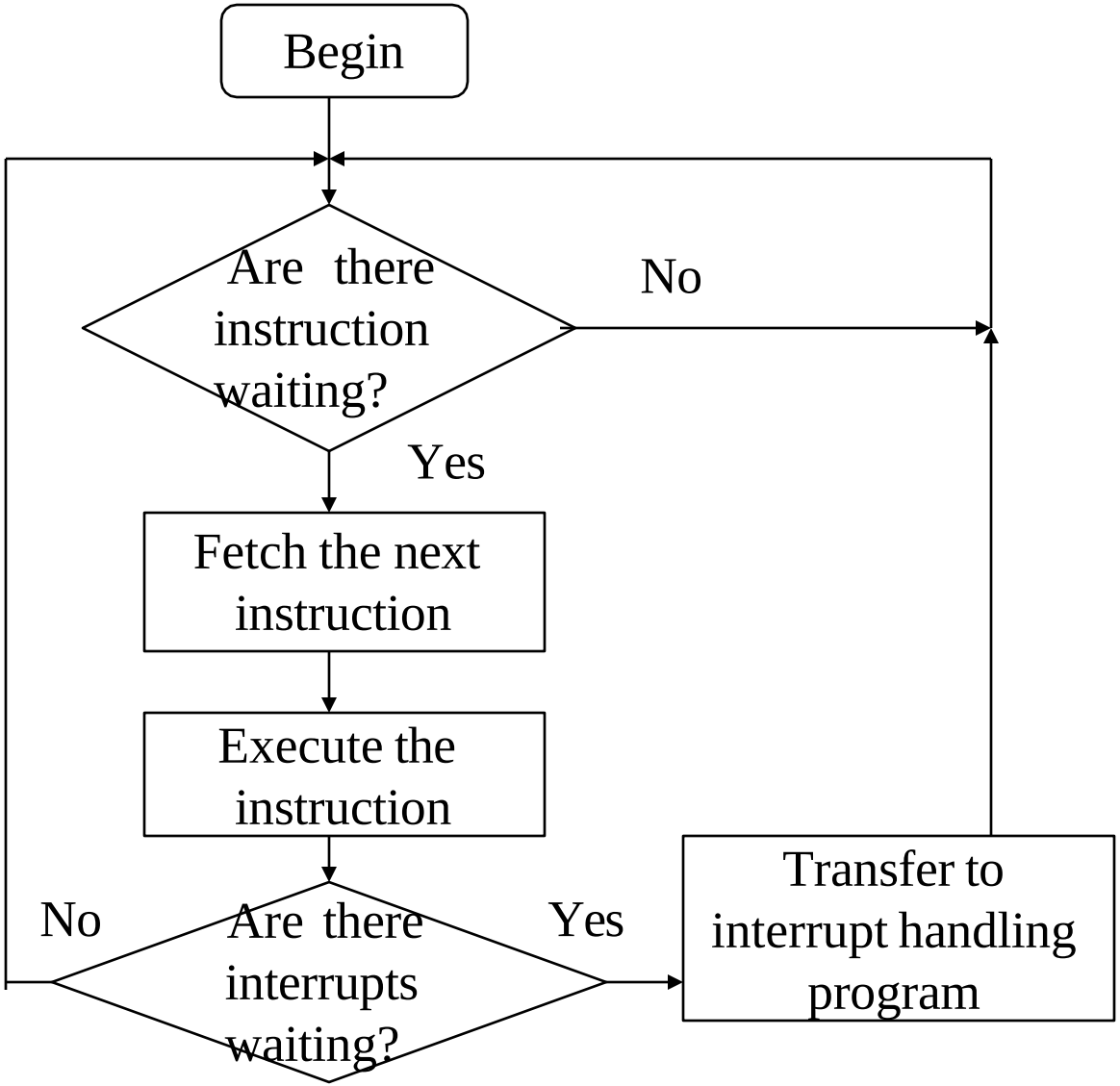
User and Supervisor Programs

- A user or application program handles a specific applications.
- A Supervisor programs manages various routine aspects of the computer system. It is typically part of the computer's operating system.

User and Supervisor Programs

- For normal operation, CPU continuously switches back and forth between user and supervisor programs.
- The requests for supervisor services from the secondary memory and IO devices are known as interrupt.
- In the event of interrupt, the CPU suspends execution of the current program and transfer to an appropriate interrupt handling program.
- CPU frequently check for the presence of interrupt request.

CPU Operation



CPU Operation

- The sequence of operations performed by the CPU in processing an instruction constitutes an **instruction cycle**.

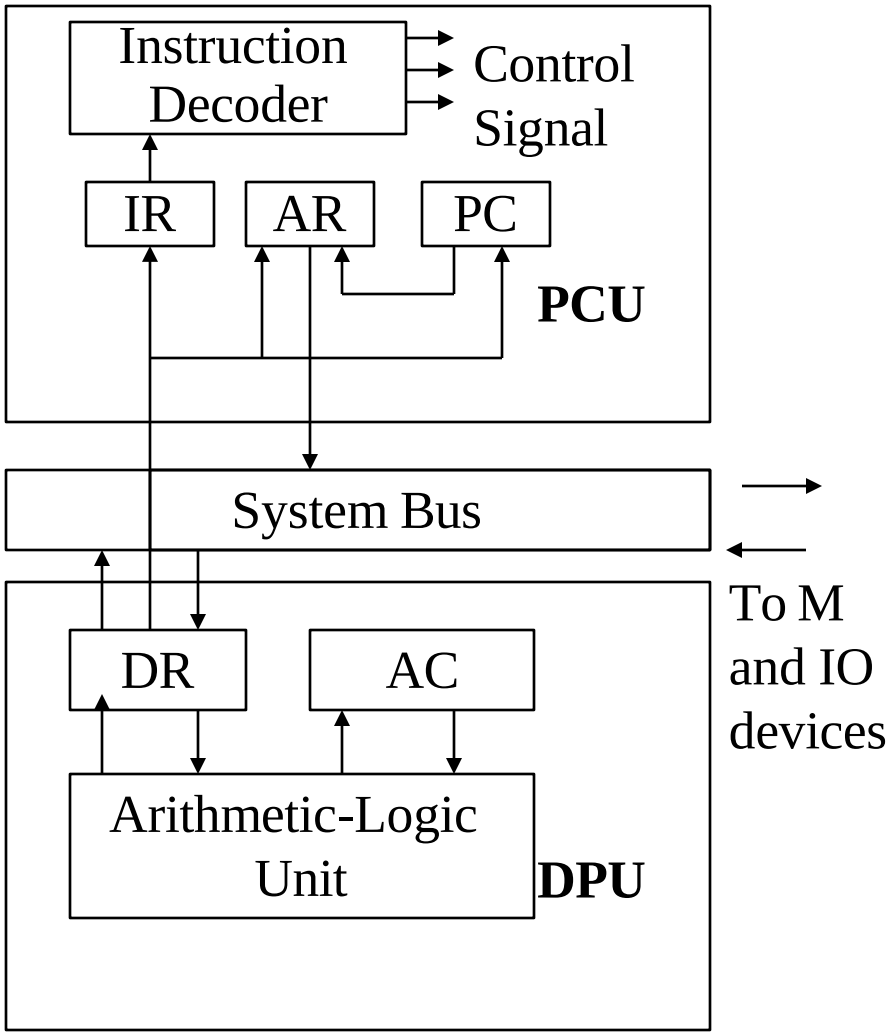
Processing of all instruction generally requires two steps:

1. **Fetch step** - a new instruction is read from the external memory.
2. **Execute step** - The operations specified by the instruction are executed.

CPU Operation

- The actions of the CPU during an instruction cycle are defined by a sequence of micro operations, each of which involves a register transfer operation.
- The time required for the shortest well-defined CPU micro operation is the CPU cycle time or clock period T_{clock}.
- The number of CPU cycles required to process an instruction varies with the instruction type.

Accumulator-based CPU



PCU = Program Control Unit

AR = Address Register

(Stores the address in memory where the CPU wants to **read/write** data)

IR = Instruction Register

(Holds the **current instruction** being decoded/executed)

PC = Program Counter

(Stores the address of the **next instruction** to be executed)

DPU = Data Processing Unit

AC = Accumulator Register

(Holds intermediate arithmetic and logic results)

DR = Data Register

(Temporarily holds the **data** read from or written to memory)

Accumulator-based CPU

- Accumulator-based architecture always stores the intermediate results of calculations in a special register, which is the accumulator. An accumulator machine is known as a 1-operand machine.

- An instruction that refers to a data word in M contains two parts, an opcode ***op*** and a memory address ***adr*** and ***I = op.adr***

- Each instruction cycle begins with the instruction fetch operation IR.AR = M (PC), where IR=***op*** and AR=***adr***. Here PC contains the address of the current instruction in the memory.

Accumulator-based CPU

- Instruction that do not reference M do not use AR. Their opcode part specifies the CPU registers to use, as well as the operation to be carried out.

- Once it has placed the opcode of I in IR, the CPU proceeds to decode and execute it. At this point, the CPU can increment PC in order to obtain the address of the next instruction.

Load and Store Operation

- The load instruction transfers a word from the memory location with address ***adr*** to the accumulator.

  AC := M (adr)

- The store instruction transfers a word from AC to M.

  M (adr) : = AC

Programming Consideration

- Data processing operations normally requires up to 3 operands. For example, Z:= X + Y.
- The accumulator based CPU supports only single-address instructions.
- A program that implement Z:=X+Y, where X, Y and Z all refer to data words in M, can take the following form:

Programming Consideration

HDL format	Assembly language format	Comment
AC:=M(X)	LD X	Load X from M into AC
DR:=AC	MOV DR, AC	Move contents of AC to DR
AC:=M(Y)	LD Y	Load Y into AC
AC:=AC+DR	ADD	Add DR to AC
M(Z):=AC	ST Z	Store contents of AC in M

HDL = Hardware Description Language
VHDL = VHSIC HDL (VHSIC = Very High-Speed Integrated Circuits)

Programming Consideration

- Consider an instruction of form: AC:= f_i (AC, M (adr)) which require to move M(adr) to or from DR and one to perform the operation f_i.
- Memory reference complicates the instruction-decoding logic.
- Overall execution time should be reduced.

Programming Consideration

HDL format	Assembly language format	Comment
AC:=M(X)	LD X	Load X from M to AC
AC:=AC+M(Y)	ADD Y	Load Y into DR and add to AC
M(Z):=AC	ST Z	Store contents of AC in M

Instruction Set

The instruction set of the accumulator based CPU:

Type	Instruction	HDL format	Assembly format
Data Transfer	Load	AC:=M(X)	LD X
	Store	M(X):= AC	ST X
	Move Register	DR:=AC	MOV DR, AC
Data Processing	Add	AC:=AC+DR	ADD
	Subtract	AC:=AC-DR	SUB
	And	AC:=AC and DR	AND
	Not	AC:= not AC	NOT
Program Control	Branch	PC:=M (adr)	BRA adr
	Branch zero	if AC = 0 then PC:=M (adr)	BZ adr

Instruction Set

The arithmetic operation negation:

HDL format	Assembly language format	Comment
DR:=AC	MOV DR, AC	Copy contents X of AC to DR
AC:=AC-DR	SUB	Compute AC = X – X = 0
AC:=AC-DR	SUB	Compute AC = 0 – X = -X

A Multiplication Program

- Consider The program should implement AC:=AC×N where, the multiplicand is the initial content of AC and the multiplier N is a variable stored in memory.
- AC×N is implemented by executing the ADD instructions N times in the form AC+AC+……..+AC
- The memory location storing N acts as a count register and after each add operation, it is decremented by 1 until it reaches 0.
- The test for N=0 is performed by means of BZ instruction.
- The memory locations:

  one – stores constant 1

  mult– store N

  ac – store Y

  prod– partial product

A Multiplication Program

Line	Location	Instruction/ Data	Assembly format
0	one	00……..01	
1	mult	N	
2	ac	00……..00	
3	prod	00……..00	
4		ST ac	
5	Loop	LD mult	AC:= M (mult)
6		BZ exit	If AC= 0 then exit
7		LD one	AC:= M (one)
8		MOV DR, AC	DR:=AC
9		LD mult	AC:= M (mult)
10		SUB	AC:=AC-DR
11		ST mult	M (mult):= AC
12		LD ac	AC:= M (ac)
13		MOV DR, AC	DR:=AC
14		LD prod	AC:= M (prod)
15		ADD	AC:=AC+DR
16		ST prod	M (prod): =AC
17		BRA Loop	
18	exit		

Several Limitations of Accumulator-based CPU

- Because of few data registers in CPU, a considerable amount of time is spent shuttling the same information back and forth between the CPU and memory.
- It would both shorten the program and speed up its execution if it is possible to store the quantities 1, N, Y and P in their own CPU registers, as they are repeatedly required by the CPU.