CO Module 3

- The CPU executes all the machine instructions and coordinates the activities of all other units during the execution of an instruction. This unit is also called as the Instruction Set Processor (ISP).
- The processor is generally called as the central processing unit (CPU)
- A typical computing task consists of a series of steps specified by a sequence of machine instructions that constitute a program.
- A program is a set of instructions performing a meaningful task. An instruction is command to the processor & is executed by carrying out a sequence of sub-operations called as micro-operations.

FUNDAMENTAL CONCEPTS

- Processor fetches one instruction at a time and perform the operation specified.
- Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.
- Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).
- Instruction Register (IR) stores currently executing instruction

EXECUTING AN INSTRUCTION

Two phases:

1. Fetch phase
2. Execution phase

- Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).

  $IR \leftarrow [[PC]]$
- Assuming that the memory is byte addressable, increment the contents of the PC by 4 (fetch phase).

  $PC \leftarrow [PC] + 4$
- Carry out the actions specified by the instruction in the IR (execution phase).

PROCESSOR ORGANIZATION

Internal organization (Functional units) of a processor contain:

- Registers for temporary storage
- Various digital circuits for executing different micro operations.(gates, MUX,decoders,counters).
- Internal path for movement of data between ALU and registers.
- Driver circuits for transmitting signals to external units.

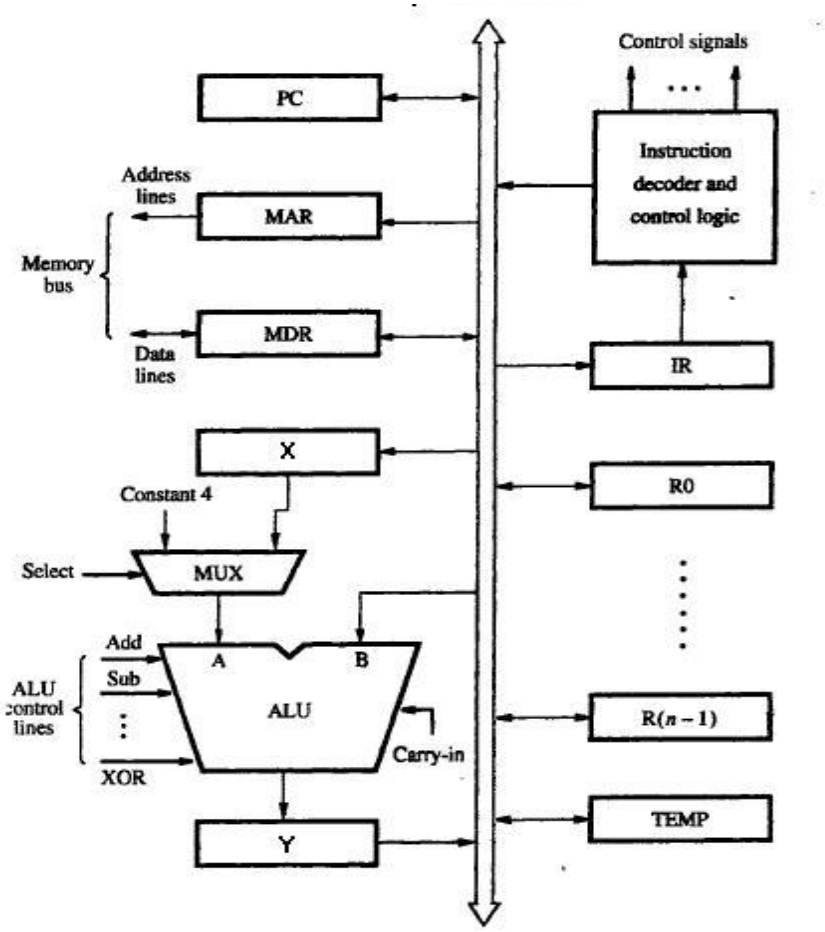
- Receiver circuits for incoming signals from external units.
- PC:
  - Keep track of execution of a program
  - Contains the memory address of the next instruction to be fetched and executed.

- MAR:
  - Holds the address of the location to be accessed.
  - I/P of MAR is connected to Internal bus and an O/p to external bus.
- MDR:

  - Contains data to be written into or read out of the addressed location.
  - IT has 2 inputs and 2 Outputs.
  - Data can be loaded into MDR either from memory bus or from internal processor bus.
- The data and address lines are connected to the internal bus via MDR and MAR
- Registers:

  - The processor registers R0 to Rn-1 vary considerably from one processor to another.
  - Registers are provided for general purpose used by programmer.
  - Special purpose registers-index & stack registers
  - Registers Y,Z &TEMP are temporary registers used by processor during the execution of some instruction.
- Multiplexer:
- Select either the output of the register Y or a constant value 4 to be provided as input A of the ALU
- Constant 4 is used by the processor to increment the contents of PC.
- ALU:
- Used to perform arithmetic and logical operation.
- Data Path:

  - The registers, ALU and interconnecting bus are collectively referred to as the data path.



Single Bus Organization

BASIC REGISTER TRANSFER

- The input and output gates for register Ri are controlled by signals isRin and Riout .
- Riin Is set to1 – data available on common bus are loaded into Ri.
- Riout Is set to1 – the contents of register are placed on the bus.
- Riout Is set to 0 – the bus can be used for transferring data from other registers

EX: Transfer the contents of R1 to R4.

- Enable output of register R1 by setting R1out=1. This places the contents of R1 on the processor bus.
- Enable input of register R4 by setting R4in=1. This loads the data from the processor bus into register R4.

Register Transfers

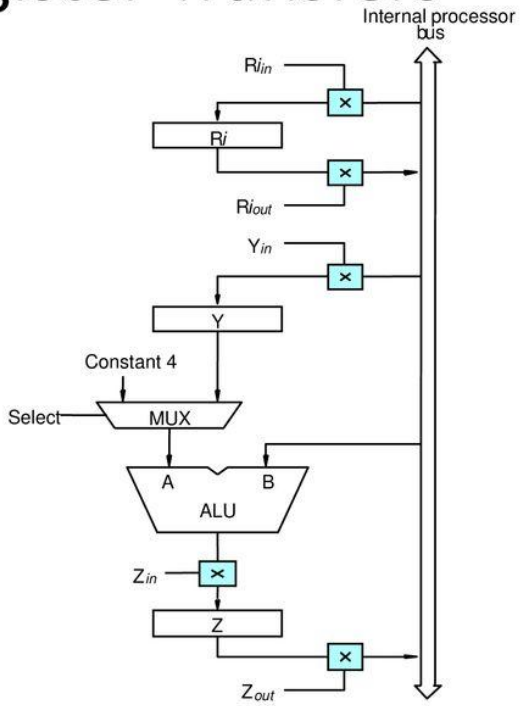


Figure 7.2. Input and output gating for the registers in Figure 7.1.

Performing an Arithmetic or Logic Operation

- The ALU is a combinational circuit that has no internal storage.
- ALU gets the two operands from MUX and bus.
- The result is temporarily stored in register Z.
- the sequence of operations to add the contents of register R1 to those of R2 and store the
  result in R3
  1. R1out, Yin
  2. R2out, SelectY, Add, Zin
  3. Zout, R3in
- Step 1: Output of the register R1 and input of the register Y are enabled, causing the contents of R1 to be transferred to Y.
- Step 2: The multiplexer's select signal is set to select Y causing the multiplexer to gate the contents of register Y to input A of the ALU.
- Step 3: The contents of Z are transferred to the destination register R3.
- All operations and data transfers are controlled by the processor clock.

Fetching a Word from Memory

CPU transfers the address of the needed information word to the memory address register (MAR). Address of the needed word is transferred to the primary memory.

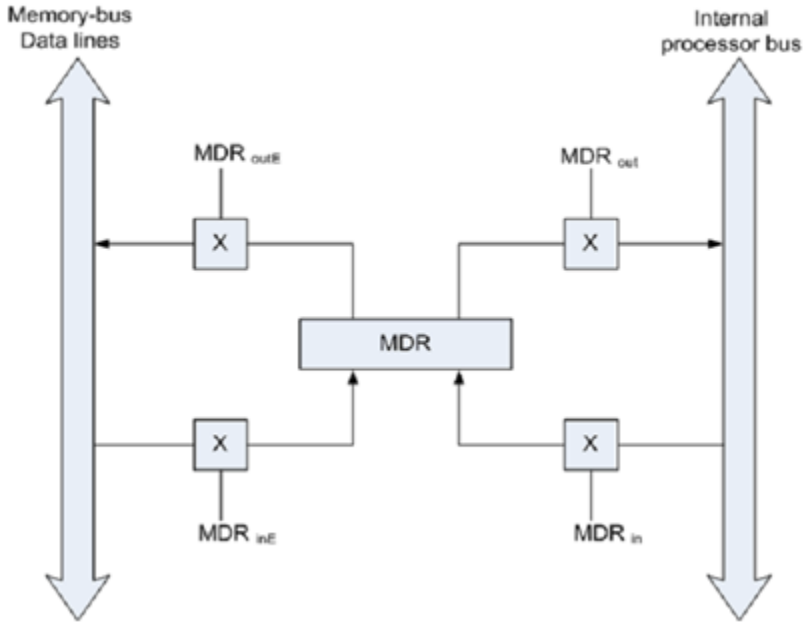
- In the meantime, the CPU uses the control lines of the memory bus to mention that a read operation is needed.
- After issuing this request, the CPU waits till it retains an answer from the memory, informing it that the required function has been finished. It is accomplished through the use of another control signal on the memory bus, which will be denoted as Memory Function Completed (MFC).
- The memory sets this signal to one to mention that the contents of the particular location in the memory have been read and are available on the data lines of the memory bus.
- We will suppose that as soon as the MFC signal is set to one, the information on the data lines is loaded into MDR and is therefore available for use inside the CPU. It finishes the memory fetch operation.

The actions required for instruction Move (R1), R2 are:

- MAR ← [R1]
- Begin Read operation on the memory bus
- Wait for the response of the MFC from the memory
- Load MDR from the memory bus
- R2 ←[MDR]

Signals activated for that problem are:

- WMFC MDRout
- R1out, MARin,
- Read MDRinE, , R2in



Storing a word in memory

That is similar process with fetching a word from memory.

- The required address is loaded into the MAR
- After that data to bewritten are loaded iinto MDR, and a write command is issued.
- If we suppose that the data word to be stored in the memory is in R2 and that the memory address is in R1, the Write operation needed the following sequence:
- MAR ←[R1]
- MDR ←[R2]
- Write
- Wait for the MFC

Move R2, (R1) requires the following sequence (signal):

- R1out, MARin
- R2out, MDRin. Write
- MDRoutE,WMFC

Execution of a Complete Instruction

We have discussed about four different types of basic operations:

- Fetch information from memory to CPU
- Store information to CPU register to memory
- Transfer of data between CPU registers.
- Perform arithmetic or logic operation and store the result in CPU registers.

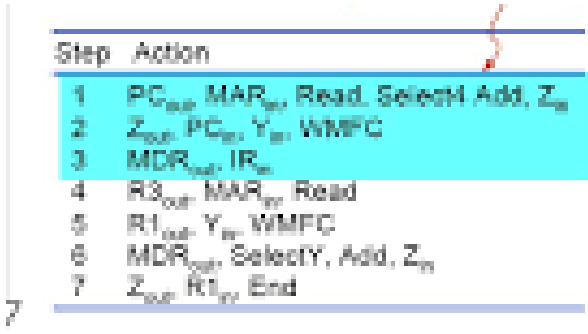
To execute a complete instruction we need to take help of these basic operations and we need to execute these operation in some particular order.

Example ADD (R1), R2

Execution of this instruction requires the following action :

1. Fetch instruction
2. Fetch first operand (Contents of memory location pointed at by the address field of the instruction)
3. Perform addition
4. Load the result into R1.

Following is the control sequence for executing a complete instruction



Step	Action
1	PC_{out}, MAR_{in}, Read, SelectH, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Step 1: The instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a Read request to the memory. The Select signal is set to Select4, which causes the multiplexer MUX to select the constant 4. This value is added to the operand at input B, which is the contents of the PC, and the result is stored in register Z.
Step 2: The updated value is moved from register Z back into the PC, while waiting for the memory to respond.
Step 3:The word fetched from the memory is loaded into the IR. (Steps 1 through 3 constitute the instruction fetch phase, which is the same for all instructions.)
Step 4: The instruction decoding circuit interprets the contents of the IR. This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the execution phase. The contents of register R3 are transferred to the MAR in step 4, and a memory read operation is initiated.
Step 5: the contents of R1 are transferred to register Y, to prepare for the addition operation.
Step 6: When the Read operation is completed, the memory operand is available in register MDR, and the addition operation is performed. The contents of MDR are gated to the bus, and thus also to the B input of the ALU, and register Y is selected as the second input to the ALU by choosing SelectY.
Step 7: The sum is stored in register Z, and then transferred to R1. The End signal causes a new instruction fetch cycle to begin by returning to step 1
(SHORT)
Instruction execution proceeds as follows:
Step1-->The instruction-fetch operation is initiated by loading contents of PC into MAR & sending a Read request to memory. The Select signal is set to Select4 ,which causes the Mux to select constant4 . This value is added to operand at input B (PC"scontent), and the result is stored in Z
Step2-->Updated value in Z is moved to PC.
Step3-->Fetched instruction is moved into MDR and then to IR.
Step4-->Contents of R3 are loaded into MAR & a memory read signal is issued.
Step5-->Contents of R1 are transferred to Y to prepare for addition.
Step6-->When Read operation is completed, memory-operand is available in MDR , and the addition is performed.

Step7-->Sum is stored in Z ,then transferred to R1. The End signal causes an ew instruction fetch cycle to begin by returning to step1.

BRANCHING INSTRUCTIONS

• Control sequence for an unconditional branch instruction is as follows:

1) PCout, MARin, Read, Select4, Add, Zin
2) Zout, PCin, Yin, WMFC
3) MDRout, IRin
4) Offset-field-of-IRout, Add, Zin
5) Zout, PCin, End

• The processing starts, as usual, the fetch phase ends in step3.
• In step 4, the offset-value is extracted from IR by instruction-decoding circuit.
• Since the updated value of PC is already available in register Y, the offset X is gated onto the bus, and an addition operation is performed.
• In step 5, the result, which is the branch-address, is loaded into the PC.
• The offset X used in a branch instruction is usually the difference between the branch target-address and the address immediately following the branch instruction. (For example, if the branch instruction is at location 1000 and branch target-address is 1200, then the value of X must be 196, since the PC will be containing the address 1004 after fetching the instruction at location 1000).
• In case of conditional branch, we need to check the status of the condition-codes before loading a new value into the PC.

e.g.: Offset-field-of-IRout, Add, Zin, If N=0 then End

If N=0, processor returns to step 1 immediately after step 4.
If N=1, step 5 is performed to load a new value into PC.

CONTROL SIGNAL GENERATION

To execute instructions, the processor must generate control signals in a proper sequence.

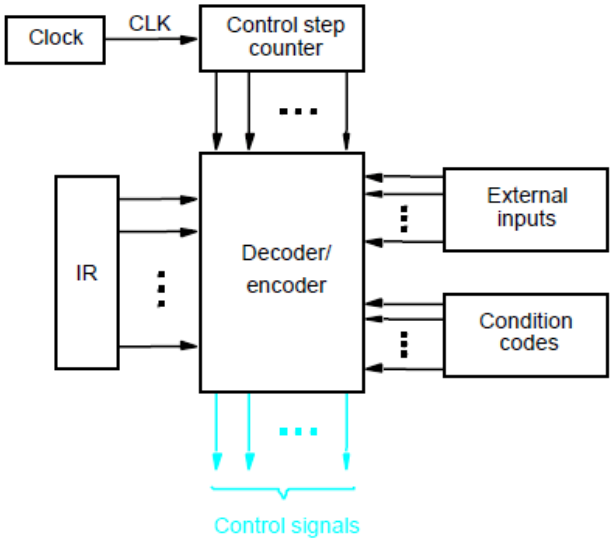
1. Hardwired control
2. Microprogrammed control

HARDWIRED CONTROL

• The control unit uses fixed logic circuits to interpret instructions and generate control signal
• Decoder/encoder block is a combinational-circuit that generates required control-outputs depending on state of all its inputs.
• A counter may be used to keep track of control steps.
• The required control signals are determined by following information

1 contents of control step counter
2 contents of instruction register
3 contents of condition code flags
4 external input signals such as MFC, interrupt request

• Step-decoder provides a separate signal line for each step in the control sequence. Similarly, output of instruction-decoder consists of a separate line for each machine instruction.
• For any instruction loaded in IR, one of the output-lines INS1 through INSm is set to 1, and all other lines are set to 0.
• The input signals to encoder-block are combined to generate the individual control-signals Yin, PCout, Add, End and so on.
• Sequence of operations carried out by this machine is determined by wiring of logic elements, hence the name “hardwired”.
• Advantage: Can operate at high speed.
Disadvantage: Limited flexibility.



MICROPROGARMMED CONTROL

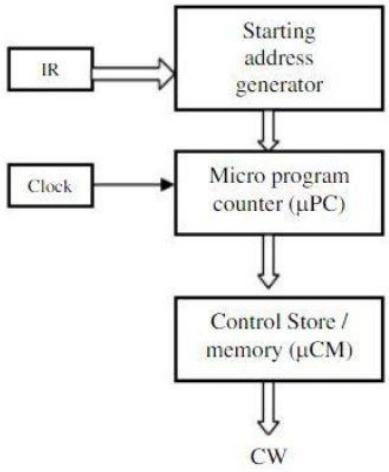


Figure 8

• Control-signals are generated by a program similar to machine language programs.
• Control word(CW) is a word whose individual bits represent various control-signals(like Add, End, Zin). {Each of the control-steps in control sequence of an instruction defines a unique combination of 1s & 0s in the CW}.
• Individual control-words in microroutine are referred to as microinstructions.
• A sequence of CWs corresponding to control-sequence of a machine instruction constitutes the microroutine.
• The microroutines for all instructions in the instruction-set of a computer are stored in a special memory called the control store(CS).
• Control-unit generates control-signals for any instruction by sequentially reading CWs of corresponding microroutine from CS.
• Microprogram counter(μPC) is used to read CWs sequentially from CS.
• Every time a new instruction is loaded into IR, output of "starting address generator" is loaded into μPC.
• Then, μPC is automatically incremented by clock, causing successive microinstructions to be read from CS. Hence, control-signals are delivered to various parts of processor in correct sequence.

COMPLETE PROCESSOR

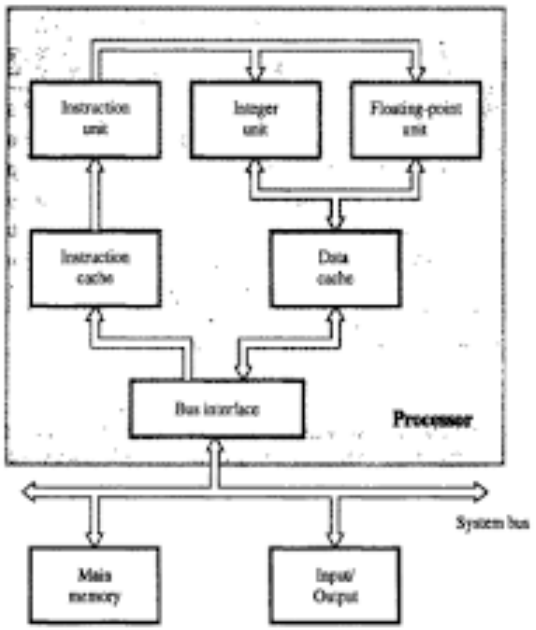


Figure 7.14 Block diagram of a complete processor.

• This has separate processing-units to deal with integer data and floating-point data.
• A data-cache is inserted between these processing-units & main-memory.
• Instruction-unit fetches instructions → from an instruction-cache or → from main-memory when desired instructions are not already in cache
• Processor is connected to system-bus & hence to the rest of the computer by means of a bus interface
• Using separate caches for instructions & data is common practice in many processors today.
• A processor may include several units of each type to increase the potential for concurrent operations.

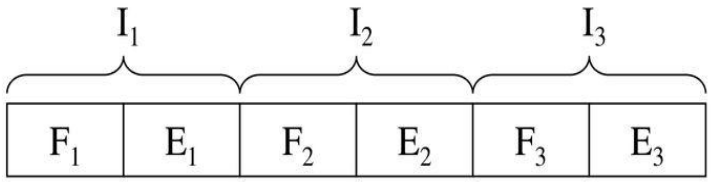
PIPELINING

Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased

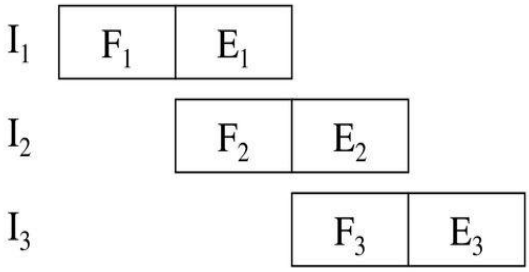
The processor executes a program by fetching and executing instructions, one after the other. Let Fi and Ei refer to the fetch and execute steps for instruction Ii .

Execution of a program consists of a sequence of fetch and execute steps shown in figure 8a consider a computer that has two separate hardware units, one for fetching instructions and another for executing them, as shown in Figure 8.1b

Pipelining



Sequential Execution



Pipelined Execution