COMPUTER ORGANIZATION AND ARCHITECTURE

MODULE 4

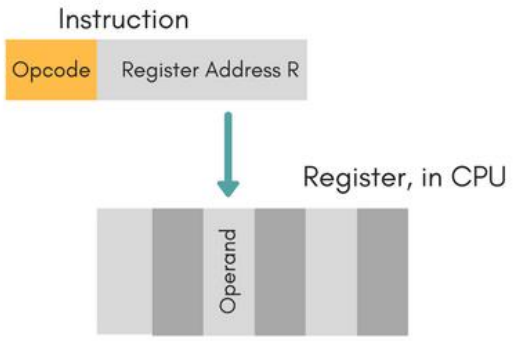
1. **What is Micro program?**
   A sequence of microinstructions constitutes a microprogram
2. **What is addressing mode and list the different types**
   The addressing mode is the method to specify the operand of an instruction.

Immediate Mode

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.
For example: ADD 7, which say Add 7 to contents of accumulator. 7 is the operand here.

Register Mode

In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.



Advantages

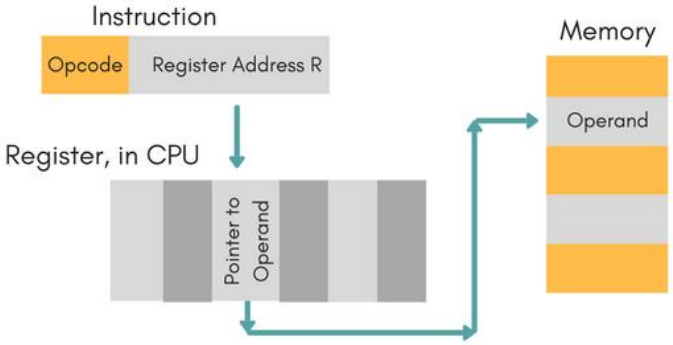
- Shorter instructions and faster instruction fetch.
- Faster memory access to the operand(s)

Disadvantages

- Very limited address space
- Using multiple registers helps performance but it complicates the instructions.

Register Indirect Mode

In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.



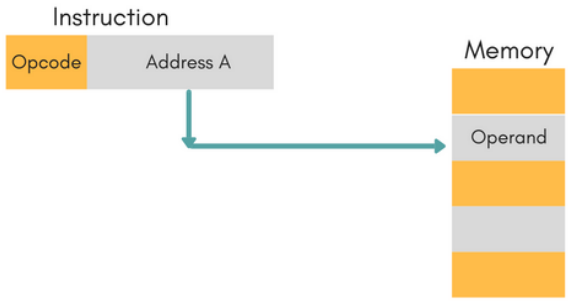
Auto Increment/Decrement Mode

In this the register is incremented or decremented after or before its value is used.

Direct Addressing Mode

In this mode, effective address of operand is present in instruction itself.

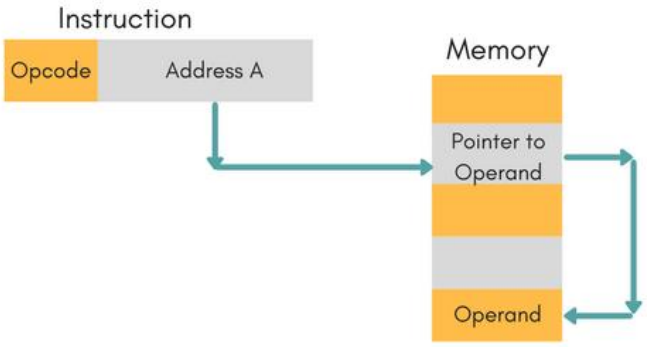
- Single memory reference to access data.
- No additional calculations to find the effective address of the operand.



For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.
NOTE: Effective Address is the location where operand is present.

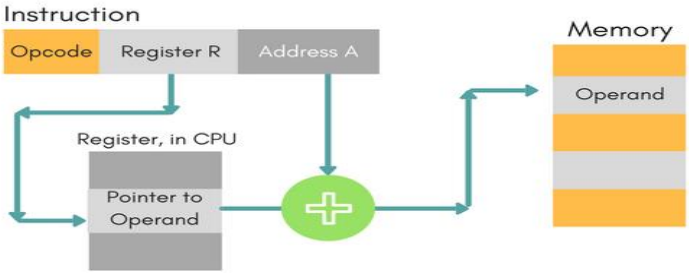
Indirect Addressing Mode

In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.



Displacement Addressing Mode

In this the contents of the indexed register is added to the Address part of the instruction, to obtain the effective address of operand.
EA = A + (R), In this the address field holds two values, A(which is the base value) and R(that holds the displacement), or vice versa.



Relative Addressing Mode

It is a version of Displacement addressing mode.
In this the contents of PC(Program Counter) is added to address part of instruction to obtain the effective address.
EA = A + (PC), where EA is effective address and PC is program counter.
The operand is A cells away from the current cell(the one pointed to by PC)

Base Register Addressing Mode

It is again a version of Displacement addressing mode. This can be defined as EA = A + (R), where A is displacement and R holds pointer to base address.

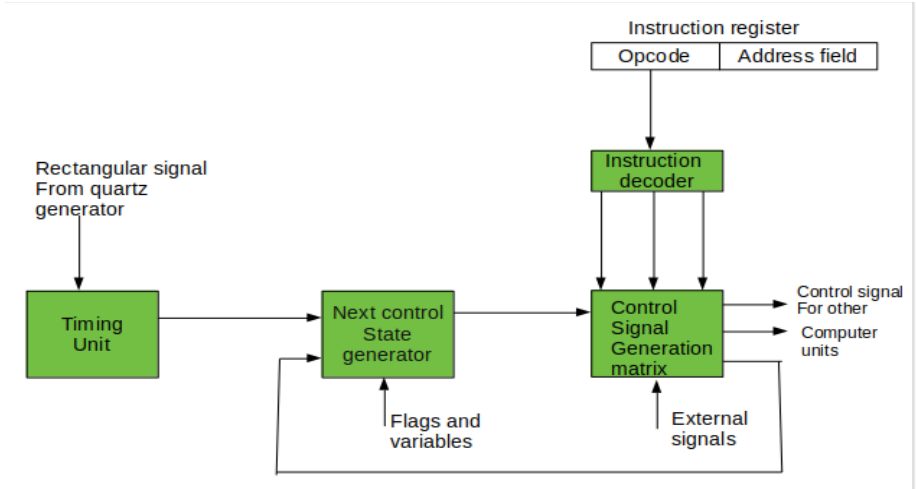
Stack Addressing Mode

In this mode, operand is at the top of the stack. For example: ADD, this instruction will *POP* top two items from the stack, add them, and will then *PUSH* the result to the top of the stack.

3. draw necesary diagrams and explain the control signal generation using hardwired control

in the Hardwired control unit, the control signals that are important for instruction execution control are generated by specially designed hardware logical circuits, in which we can not modify the signal generation method without physical change of the circuit structure. The operation code of an instruction contains the basic data for control signal generation. In the instruction decoder, the operation code is decoded. The instruction decoder constitutes a set of many decoders that decode different fields of the instruction opcode.

As a result, few output lines going out from the instruction decoder obtains active signal values. These output lines are connected to the inputs of the matrix that generates control signals for executive units of the computer. This matrix implements logical combinations of the decoded signals from the instruction opcode with the outputs from the matrix that generates signals representing consecutive control unit states and with signals coming from the outside of the processor, e.g. interrupt signals. The matrices are built in a similar way as a programmable logic arrays.



Block diagram of a hardwired control unit of a computer

Control signals for an instruction execution have to be generated not in a single time point but during the entire time interval that corresponds to the instruction execution cycle. Following the structure of this cycle, the suitable sequence of internal states is organized in the control unit.

A number of signals generated by the control signal generator matrix are sent back to inputs of the next control state generator matrix. This matrix combines these signals with the timing signals, which are generated by the timing unit based on the rectangular patterns usually supplied by the quartz generator. When a new instruction arrives at the control unit, the control units is in the initial state of new instruction fetching. Instruction decoding allows the control unit enters the first state relating execution of the new instruction, which lasts as long as the timing signals and other input signals as flags and state information of the computer remain unaltered. A change of any of the earlier mentioned signals stimulates the change of the control unit state.

This causes that a new respective input is generated for the control signal generator matrix. When an external signal appears, (e.g. an interrupt) the control unit takes entry into a next control state that is the state concerned with the reaction to this external signal (e.g. interrupt processing). The values of flags and state variables of the computer are used to select suitable states for the instruction execution cycle.

The last states in the cycle are control states that commence fetching the next instruction of the program: sending the program counter content to the main memory address buffer register and next, reading the instruction word to the instruction register of computer. When the ongoing instruction is the stop instruction that ends program execution, the control unit enters an operating system state, in which it waits for a next user directive.

4. Write in detail about program control.

PROGRAM CONTROL INSTRUCTIONS :

- It is sometimes convenient to supplement the ALU circuit in the CPU with a status register where status bit conditions be stored for further analysis. Status bits are also called condition-code bits or flag bits.
- Figure 5.3 shows the block diagram of an 8-bit ALU with a 4-bit status register. The four status bits are symbolized by C, S, Z, and V. The bits are set or cleared as a result of an operation performed in the ALU.

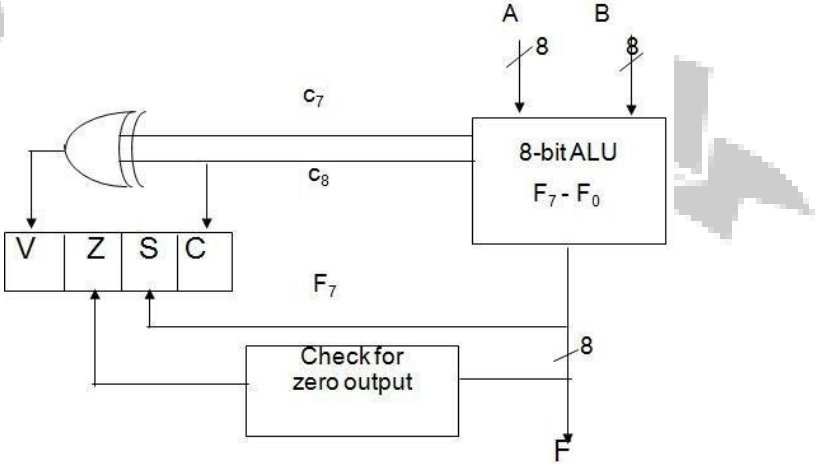


Figure 5.3: Status Register Bits

1. Bit C (carry) is set to 1 if the end carry C8 is 1. It is cleared to 0 if the carry is 0.
2. Bit S (sign) is set to 1 if the highest-order bit F7 is 1. It is set to 0 if set to 0 if the bit is 0.
3. Bit Z (zero) is set to 1 if the output of the ALU contains all 0's. it is cleared to 0 otherwise. In other words, Z = 1 if the output is zero and Z = 0 if the output is not zero.
4. Bit V (overflow) is set to 1 if the exclusives-OR of the last two carries is equal to 1, and cleared to 0 otherwise. This is the condition for an overflow when negative numbers are in 2's complement. For the 8-bit ALU, V = 1 if the output is greater than + 127 or less than -128.

- The status bits can be checked after an ALU operation to determine certain relationships that exist between the vales of A and B.
- If bit V is set after the addition of two signed numbers, it indicates an overflow condition.
- If Z is set after an exclusive-OR operation, it indicates that A = B.
- A single bit in A can be checked to determine if it is 0 or 1 by masking all bits except the bit in question and then checking the Z status bit.

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RTN
Compare(by -)	CMP
Test (by AND)	TST

CMP and TST instructions do not retain their results of operations(- and AND, respectively). They only set or clear certain Flags.

Subroutine call and return

Call subroutine
Jump to subroutine
Branch to subroutine
Brach and save return address

PROGRAM INTERRUPT:

There are three major types of interrupts that cause a break in the normal execution of a program. They can be classified as:

1. External interrupts
2. Internal interrupts
3. Software interrupts

1) External interrupts:

- External interrupts come from input-output (I/0) devices, from a timing device, from a circuit monitoring the power supply, or from any other external source.

2) Internal interrupts:

- Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called traps.

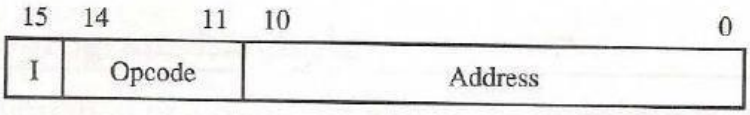
3) Software interrupts:

- A software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program.

5. **interrupt initiated by an instruction is called as -----**
   External
6. **A stack organized computer uses instruction of ---- addressing**
   Zero addressing
7. **Write a short note on instruction format.**

OP-code field - specifies the operation to be performed
Address field - designates memory address(s) or a processor register(s)
Mode field - specifies the way the operand or the effective address is determined.
The number of address fields in the instruction format depends on the internal organization of CPU

The computer instruction format is shown in below figure.



(a) Instruction format

Three fields for an instruction:

i. 1-bit field for indirect addressing
ii. 4-bit opcode
iii. 11-bit address field

The example will only consider the following 4 of the possible 16 memory instructions

Symbol	Opcode	Description
ADD	0000	$AC \leftarrow AC + M[EA]$
BRANCH	0001	If $(AC < 0)$ then $(PC \leftarrow EA)$
STORE	0010	$M[EA] \leftarrow AC$
EXCHANGE	0011	$AC \leftarrow M[EA], M[EA] \leftarrow AC$

EA is the effective address

(b) Four computer instructions

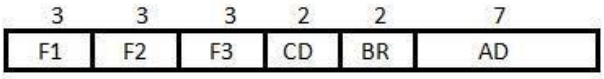
8. **Write a short note on micro instruction format.**

Microinstruction Format

The microinstruction format for the control memory is shown in figure 4.5. The 20 bits of the microinstruction are divided into four functional parts as follows:

1. The three fields F1, F2, and F3 specify microoperations for the computer.
   The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations. This gives a total of 21 microoperations.
2. The CD field selects status bit conditions.
3. The BR field specifies the type of branch to be used.
4. The AD field contains a branch address. The address field is seven bits

wide, since the control memory has 128 = 2^7 words.



F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

9. Explain data transfer and manipulation instruction.

DATA TRANSFER INSTRUCTIONS

Typical Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Data Transfer Instructions with Different Addressing Modes

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	AC ← M[ADR]
Indirect address	LD @ADR	AC ← M[M[ADR]]
Relative address	LD $ADR	AC ← M[PC + ADR]
Immediate operand	LD #NBR	AC ← NBR
Index addressing	LD ADR(X)	AC ← M[ADR + XR]
Register	LD R1	AC ← R1
Register indirect	LD (R1)	AC ← M[R1]
Autoincrement	LD (R1)+	AC ← M[R1], R1 ← R1 + 1
Autodecrement	LD -(R1)	R1 ← R1 - 1, AC ← M[R1]

DATA MANIPULATION INSTRUCTIONS

Three Basic Types: Arithmetic instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with Borrow	SUBB
Negate(2's Complement)	NEG

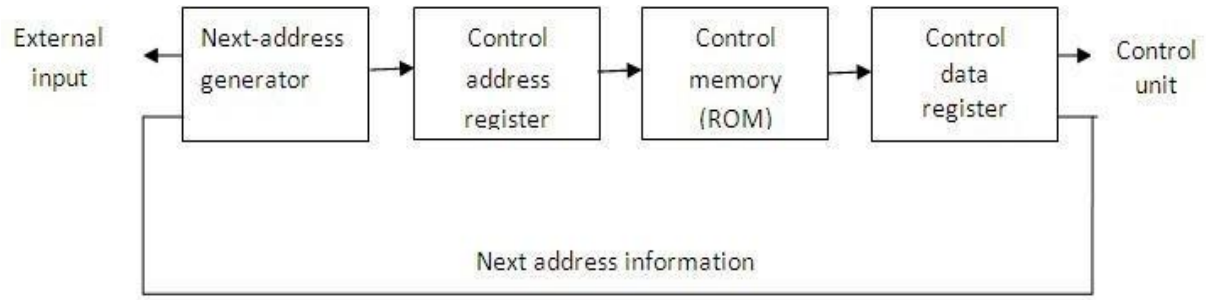
Logical and Bit Manipulation Instructions Shift Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right thru carry	RORC
Rotate left thru carry	ROLC

10. Explain Organization of micro programmed control unit

- The control memory is assumed to be a ROM, within which all control information ispermanently stored.



Micro-programmed control organization

- The control memory address register specifies the address of the microinstruction, and the control data register holds the microinstruction read from memory.
- The microinstruction contains a control word that specifies one or more microoperations for the data processor. Once these operations are executed, the control must determine the next address.
- The location of the next microinstruction may be the one next in sequence, or it may belocated somewhere else in the control memory.
- While the microoperations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.
- Thus a microinstruction contains bits for initiating microoperations in the data processor part and bits that determine the address sequence for the control memory.
- The next address generator is sometimes called a ***micro-program sequencer***, as itdetermines the address sequence that is read from control memory.
- Typical functions of a micro-program sequencer are incrementing the control address register by one, loading into the control address register an address from control memory, transferring an external address, or loading an initial address to start the control operations.
- The control data register holds the present microinstruction while the next address iscomputed and read from memory.
- The data register is sometimes called a ***pipeline register***.
- It allows the execution of the microoperations specified by the control word simultaneously with the generation of the next microinstruction.
- This configuration requires a two-phase clock, with one clock applied to the address register and the other to the data register.
- The main advantage of the micro programmed control is the fact that once the hardware configuration is established; there should be no need for further hardware or wiring changes.

11. Explain Address Sequencing

Step-1:

- An initial address is loaded into the control address register when power

is turned on inthe computer.

- This address is usually the address of the first microinstruction that activates the instruction fetch routine.
- The fetch routine may be sequenced by incrementing the control address register through the rest of its microinstructions.
  - At the end of the fetch routine, the instruction is in the instruction register of the computer

Step-2:

- The control memory next must go through the routine that determines the effective address of the operand.
- A machine instruction may have bits that specify various addressing modes, such as indirect address and index registers.
- The effective address computation routine in control memory can be reached through a branch microinstruction, which is conditioned on the status of the mode bits of the instruction.
- When the effective address computation routine is completed, the address of the operand is available in the memory address register.

Step-3:

- The next step is to generate the microoperations that execute the instruction fetched from memory.
- The microoperation steps to be generated in processor registers depend on the operation code part of the instruction.
- Each instruction has its own micro-program routine stored in a given location of control memory.
- The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a ***mapping*** process.
- A mapping procedure is a rule that transforms the instruction code into a control memory address.

Step-4:

- Once the required routine is reached, the microinstructions that execute the instruction may be sequenced by incrementing the control address register.
- Micro-programs that employ subroutines will require an external register for storing the return address.
- Return addresses cannot be stored in ROM because the unit has no writing capability.
- When the execution of the instruction is completed, control must return to the fetch routine.
- This is accomplished by executing an unconditional branch microinstruction to the first address of the fetch routine.

12. Explain the Design of control Unit:

- The control memory out of each subfield must be decoded to provide the distinct microoperations.
- The outputs of the decoders are connected to the appropriate inputs in the processor unit.
- The below figure shows the three decoders and some of the connections that must be

made from their outputs.

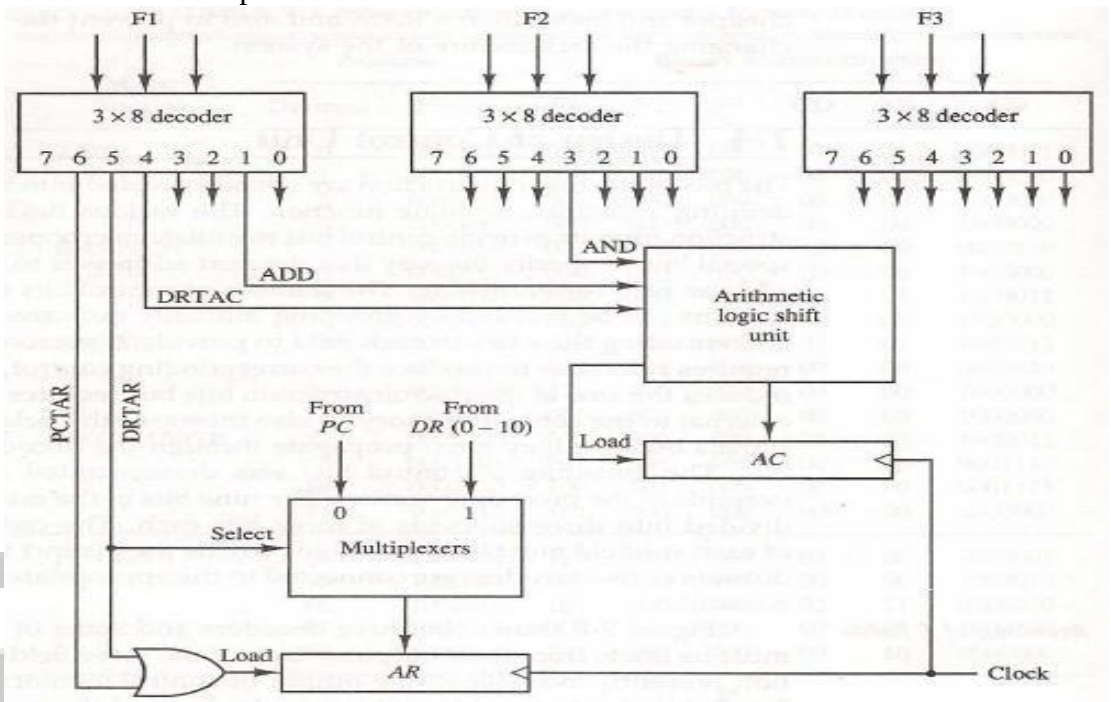


Figure 7-7 Decoding of microoperation fields.

- The three fields of the microinstruction in the output of control memory are decoded with a 3x8 decoder to provide eight outputs.
- Each of the output must be connected to proper circuit to initiate the corresponding microoperation as specified in previous topic.
- When F1 = 101 (binary 5), the next pulse transition transfers the content of DR (0-10) to AR.
- Similarly, when F1= 110 (binary 6) there is a transfer from PC to AR (symbolized by PCTAR). As shown in Fig, outputs 5 and 6 of decoder *F1* are connected to the load input of AR so that when either one of these outputs is active, information from the multiplexers is transferred to AR.
- The multiplexers select the information from DR when output 5 is active and from PC when output 5 is inactive.
- The transfer into AR occurs with a clock transition only when output 5 or output 6 of the decoder is active.
- For the arithmetic logic shift unit the control signals are instead of coming from the logical gates, now these inputs will now come from the outputs of AND, ADD and DRTAC respectively.