Subject: DECO
Faculty: A. Swathi
Topic: Introduction to Computer organization

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No. of SP
Book Reference:
Date Conducted:
Page No: 1

Computer organization

Computer organization and Architecture is used to design computer system.

Computer Architecture: It is considered

Computer Architecture refers to the end to end structure of a computer system that determines how its components interact with eachother in helping to execute the machine's purpose (i.e, processing data), often avoiding any reference to the actual technical implementation.

- Computer Architecture defines the system in an abstract manner, it deals with what does the system do.

Computer organization:

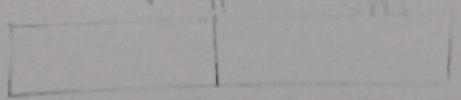
Computer organization is realization of what is specified by the computer architecture.

- It deals with how operational attributes are linked together to meet the requirements specified by computer architecture.
- Some organizational attributes are
  hardware details
  Control signals and
  Peripherals.

The basic minimum features of all digital computers contains the following components:

(1) Internal registers
(2) The main memory
(3) A set of instruction
(4) The timing and control structure.

- A program is a set of instructions that specify the operations, operands and the sequence by which processing has to occur.
- An instruction is a binary code that specifies a sequence of micro operations. (or)
  A computer instruction refers to a binary code that controls how a computer performs micro-operations in a series.
- Instructions together with the information', are saved in memory. i.e instructions and data are stored in memory.
- Every computer has its own set of instructions.
- The bits of an instruction can be grouped into parts called fields.



Subject: DECO
Faculty: A. Swathi
Topic: Instruction Code

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No... of SP
Book Reference:
Date Conducted:
Page No: 2

Instruction Code

An instruction is a command given to a computer to perform an operation on data.

- An instruction code is a group of bits that instruct the computer to perform a specific operation.

The operation code of an instruction is a group of bits that define operations such as addition, subtraction, shift, complement etc.

- An instruction consists of 3 parts

these three parts of an instruction are called fields.

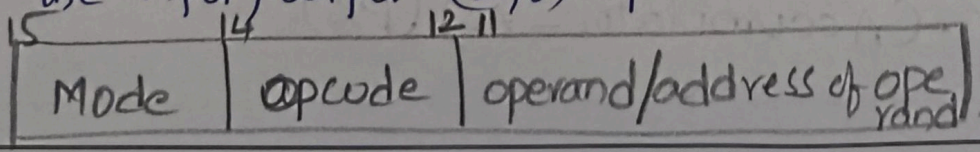
- In a computer, the instruction format usually consists of 3 fields.

These are

operation code (or) opcode field: The value of opcode specifies the operation, such as addition, subtraction.... etc to be performed on the operands.

Mode field: An I bit, which is a single bit specifying the addressing mode

Address field: Address field specifies the address of the main memory or the register reference operation or the input/output (I/O) operation.



Computer Registers :

Registers are temporary storage units of a computer that keep both data and instructions in a binary form.

- Registers are electronic device that temporarily hold values in the form of 1 and 0.
- In computer organisation, the registers are a type of computer memory used to accept, store and transfer data and instructions used by the cpu right way.
- During the execution of program, registers are used to store data temporarily.

Types of Registers :

1. one register for holding data called - DataRegister(DR)
2. one register for storing instruction called - Instruction Register (IR)
3. A register for holding and address of memory word called - Address Register (AR)
4. A register for holding temporary data generated during processing. This register is named as - Temporary Register. (TR)
5. A register is required for doing operations on data. This processor register holds data on which addition, subtraction, multiplication, shift and logical operation are to be carried out - A processor Register (Accumulator - AC) (PR)

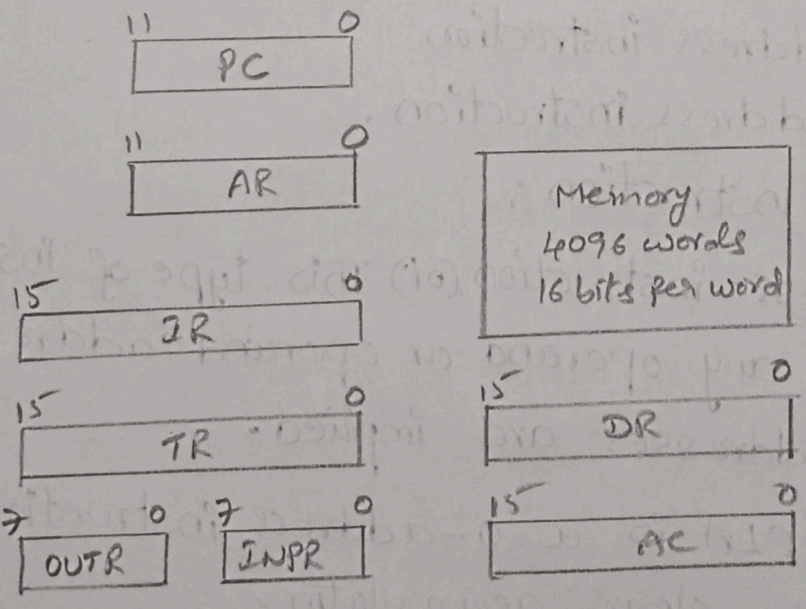
Subject: DECO
Faculty: A. Swathi
Topic: Registers

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:

6. A register that will act as a counter and will hold the address of next instruction. such a register is named as - program counter (PC)

7. Registers for inputting and outputting data.
INPR Input Register will hold data obtained from user through input devices like keyboard.
OUTR output Register will hold data that need to be sent to output devices like monitor, printer etc.



Basic Computer registers and Memory

A computer instruction refers to a binary code that controls how a computer performs micro-operations in a series.

Every computer has its own set of instructions. In computer operation codes or opcodes and addresses are the two elements that they are divided into.

- Instructions are of different types depending on the number of operands they contain or require, to operate on.

These are.

1. zero-address instruction
2. One-address instruction
3. Two-address instruction
4. Three-address instruction.

zero-address instruction:

zero-address instruction (or) This type of instructions do not contain any operand or operand address.

- The operand addresses are implied.
- The instruction CLA is a 0-address instruction.
- CLA stands for clear accumulator.
- Here the instruction itself specifies that the operation "clear" is to be performed on Accumulator.

Subject: DCCo
Faculty: A. Swathi
Topic: Computer Instructions and Instruction Formats

Class Notes

Unit No: IV
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 4

One-address instruction:

In this type of instruction, a single operand address is specified. The other operand lies in the accumulator and the result is also stored back in the accumulator.

Ex: Some examples of 1-address instruction.

```
LDA   A
STA   A
ADD   A.
```

Two-address instruction:

In this type of instructions, addresses of two operands are specified. The result of the operation is stored in one of the given operand addresses.

Ex:
```
MOV   R1, R2
ADD   R1, R2
```

Three-address instruction:

In three address instructions, two addresses are specified and a third address is provided for storing the result.

Ex: ADD R1, A, B

Leads to addition of operands at locations A and B and the sum of the two operands are stored in register R_1.

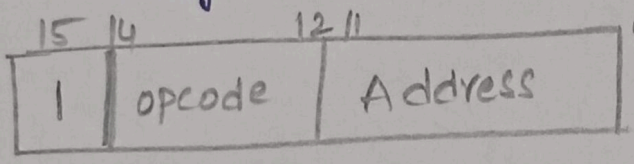
Instruction Code formats (COMPUTER INSTRUCTIONS):

A basic computer has three instruction code formats which are

1. Memory-reference instruction
2. Register-reference instruction
3. Input-output instruction.

Memory reference instruction:

In memory reference instruction, 12 bits of memory is used to specify an address and one bit to specify the addressing mode 'I'.

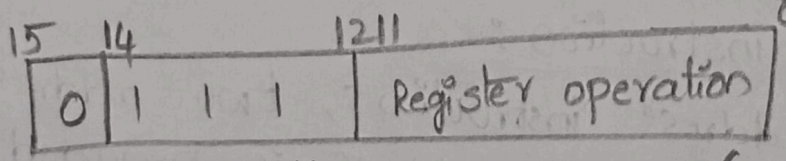


(opcode = 000 through 110)

Register reference instruction:

In register reference instructions are represented by the opcode 111 with a 0 in the leftmost bit (bit 15) of the instruction.

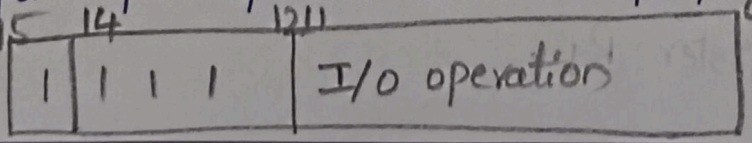
A register reference instruction specifies an operation on or a test of the AC (Accumulator) register.



(opcode = 111, I=0)

Input-output instruction:

Just like register reference instruction, an input-output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the left most bit of the instruction. The remaining 12 bits are used to specify the type of input-output operation or test performed.



Subject: DECO
Faculty: A. Swathi
Topic: Addressing Modes

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 5

Addressing Modes:

Instructions that define the address of a definite memory location are known as memory reference instructions.
The method in which a target address or effective address is recognized with in the instruction is known as addressing mode.
The address field for instruction can be represented in two different ways are as follows.

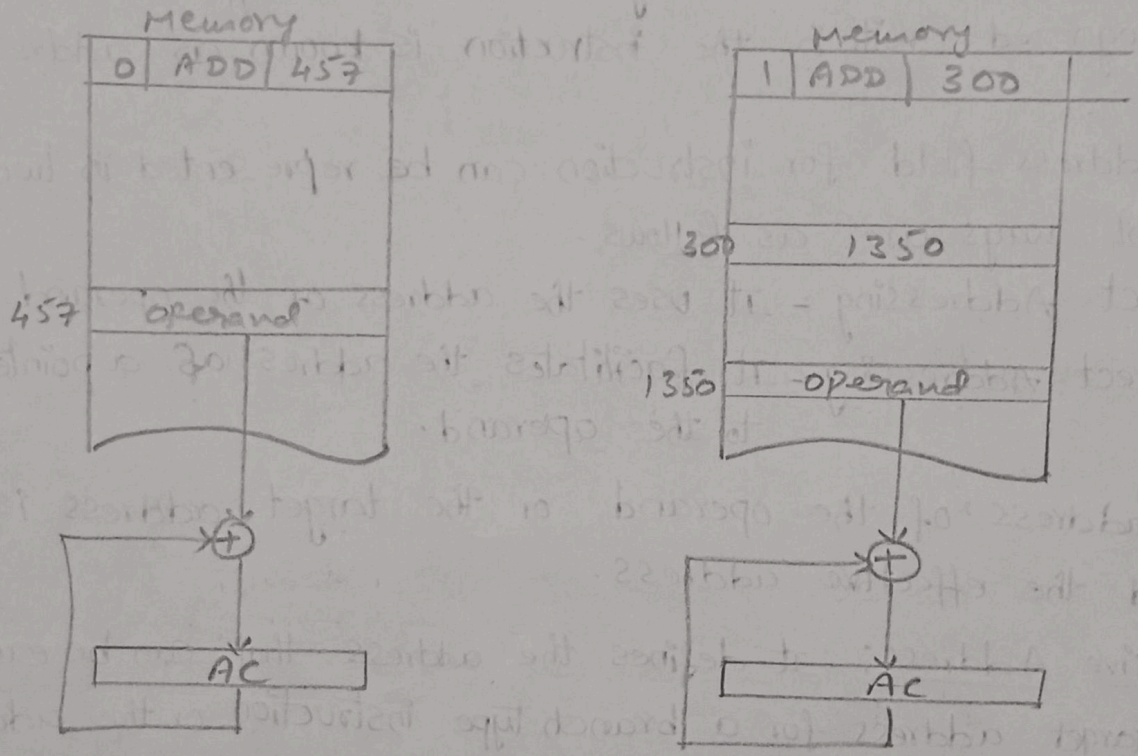
1. Direct Addressing - It uses the address of the operand.
2. Indirect Addressing - It facilitates the address of a pointer to the operand.

The address of the operand or the target address is called the effective address.

Effective Address: It defines the address that can be executed as a target address for a branch type instruction or the address that can be used directly to create an operand for a computation type instruction, without creating any changes.

Address:
The address is represented as the locator where a specific instruction is constructed in the memory.
The address bits of an instruction code is used as an operand and not as an address.
In such methods, the instruction has an immediate operand.
If the second part has an address, the instruction is referred to have a direct address.

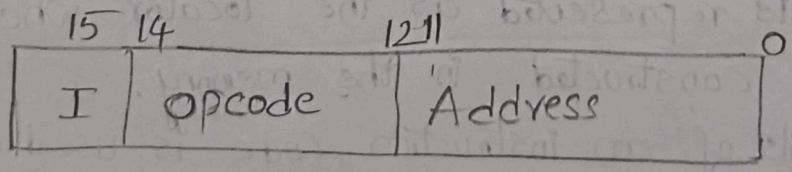
• There is another possibility in the second part including the address of operand. This is referred to as an indirect address. In the instruction code, one bit can signify if the direct or indirect address is executed.



Direct Address

Indirect Address.

Demonstration of Direct and Indirect Address



Instruction format

Subject: DECO
Faculty: A. Swathi
Topic: Timing and Control

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 6

Timing and control

The timing for all registers in the basic computer is controlled by a master clock generator.

- The clock pulses are applied to all flipflops and registers in the system, including the flipflops and registers in the control unit.

The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers and microoperations for the accumulator.

There are two major types of control organization:

1. Hardwired control:

The control logic is implemented with gates, flip-flops, decoders, and other digital circuits.

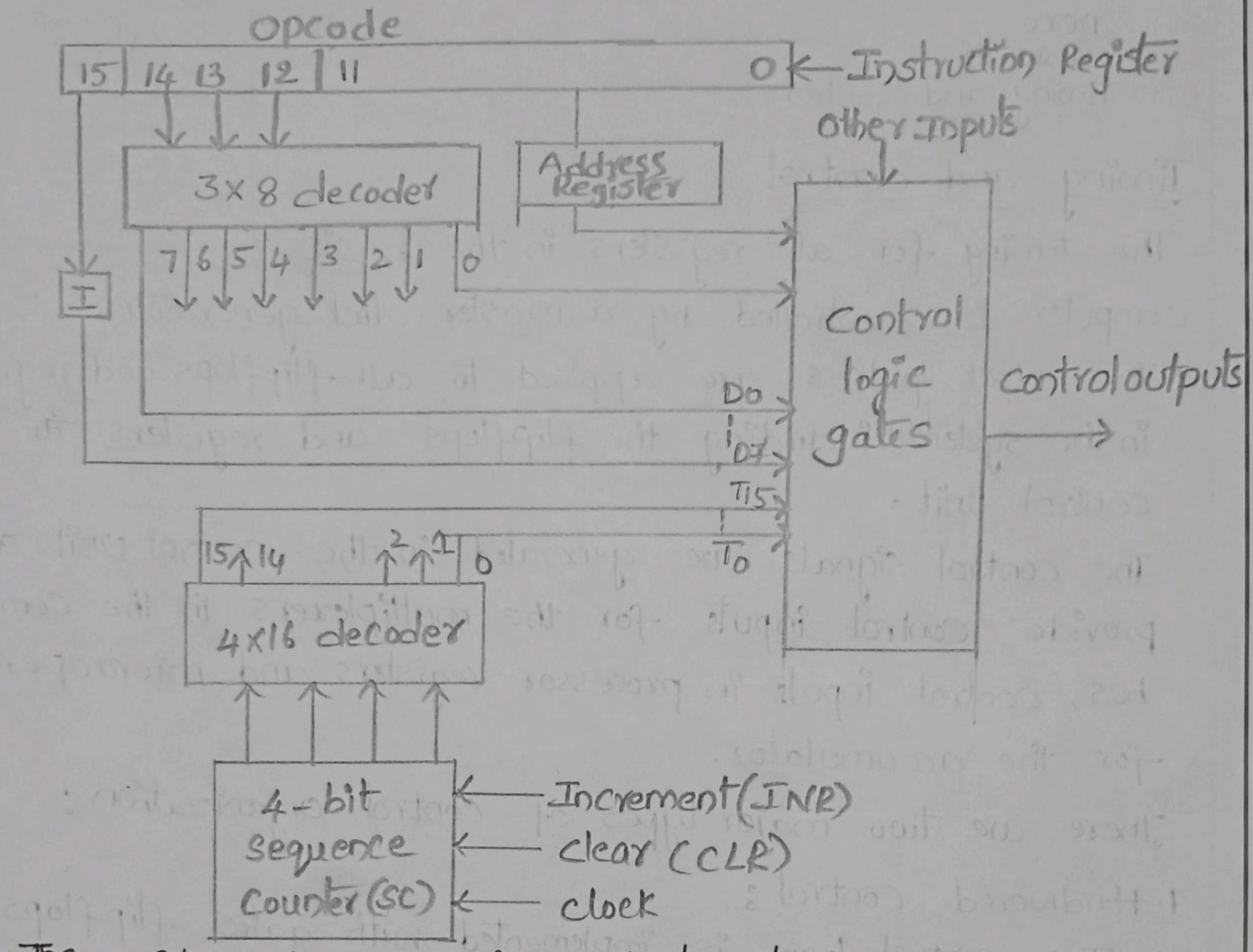
It has the advantages that it can be optimized to produce a fast mode of operation.

2. Microprogrammed control:

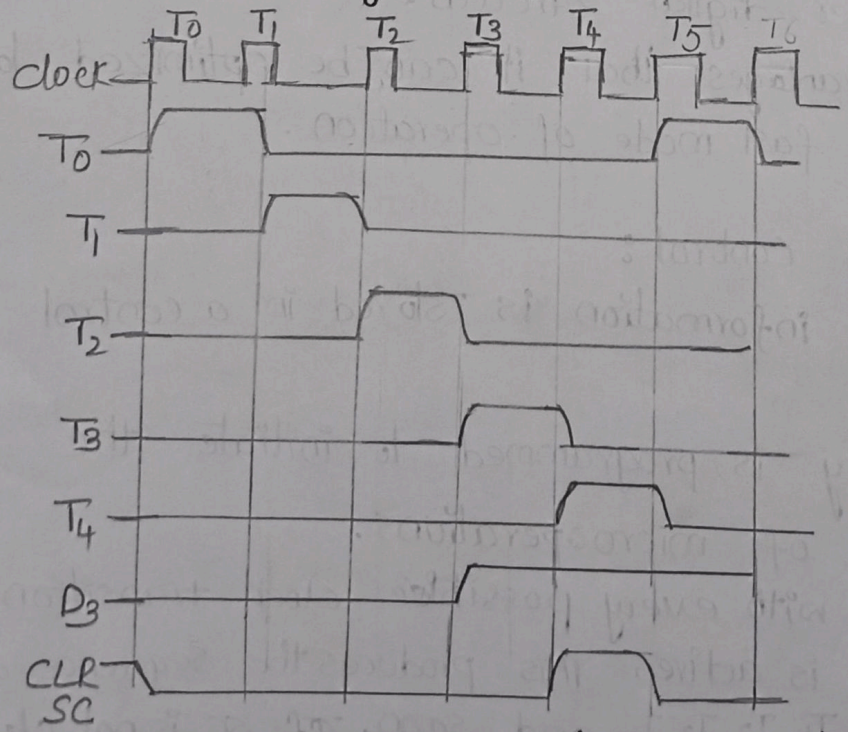
The control information is stored in a control memory.

The control memory is programmed to initiate the required sequence of microoperations.

- SC is incremented with every possible clock transition, unless is CLR input is active. This produces the sequence of timing signals T_0, T_1, T_2, T_3, T_4 and so on, If SC is not cleared, the timing signals with continue with T_5, T_6 upto T_{15} and back to T_0.



The Block diagram of control unit



Example of control timing signals.

Subject: DECO
Faculty: A. Swathi
Topic: Instruction cycle.

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 7

Instruction cycle

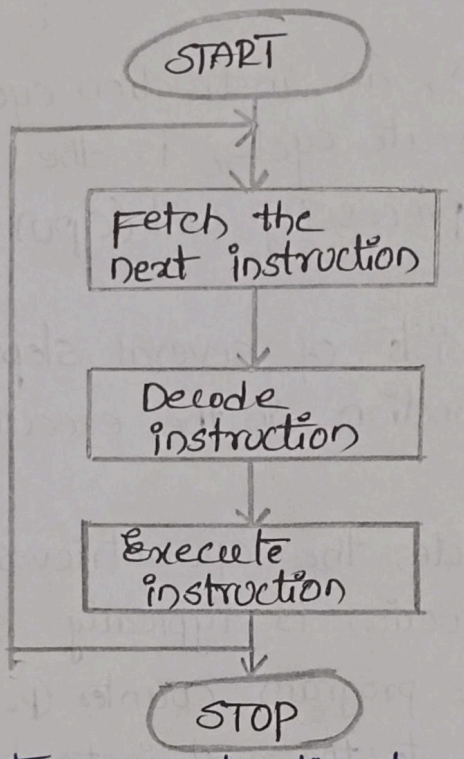
In computer organization, an instruction cycle, also known as a fetch-decode-execute cycle, is the basic operation performed by a central processing unit (CPU) to execute an instruction.

The instruction cycle consists of several steps, each of which performs a specific function in the execution of the instruction.

1. Fetch: In the fetch cycle, the CPU retrieves the instruction from memory. The instruction is typically stored at the address specified by the program counter (PC). The PC is then incremented to point to the next instruction in memory.
2. Decode: In the decode cycle, the CPU interprets the instruction and determines what operation needs to be performed. This involves identifying the opcode and any operands that are needed to execute the instruction.
3. Execute: In the execute cycle, the CPU performs the operation specified by the instruction. This may involve reading or writing data from or to memory, performing arithmetic or logic operations on data, or manipulating the control flow of the program.

- These cycles are the basic building blocks of the CPU's operation and are performed for every instruction executed by the CPU.

By optimizing these cycles, cpu designers can improve the performance and efficiency of the cpu, allowing it to execute instructions faster and more effeciently.



The advantages and disadvantages of the instruction cycle depend on various factors, such as the specific cpu architecture and the instruction set used.

Advantages of instruction cycle

1. standardization
2. Efficiency
3. pipelining

Disadvantages of instruction cycle

1. overhead
2. Complexity
3. Limited parallelism.

Subject: DECO
Faculty: A. Swathi
Topic: Memory Reference Instructions

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 8

- This process continues indefinitely unless a HALT instruction is encountered, Initially, the program counter PC is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoded timing signal T_0. After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on.
- The microoperations for the fetch and decode phases can be specified by the following register transfer statements.

T_0 : AR ← PC

T_1 : IR ← M[AR], PC = PC + 1

T_2 : $D_0, \ldots D_7$ ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)

It is necessary to use timing signal T_1 to provide the following connections in the bus system.

1. Enable the read input of memory.
2. place the content of memory onto the bus by making $S_2 S_1 S_0 = 111$.
3. Transfer the content of the bus to IR by enabling the LD input of IR
4. Increment PC by enabling the INR input of PC.

Memory Reference Instructions :

Memory reference instructions are those instructions where the operand to be operated upon is fetched from the memory.

We have different memory reference instruction to a basic computer.

We shall use these symbols as mnemonics in assembly language programs later.

List of memory Reference Instructions.

Symbol	Description.
AND A	- Do the logic AND operation on data stored at location A with the contents of AC.
ADD A	- A + contents of AC
LDA A	- load operand at A to AC
STA A	- store value of AC at location A
BUN A	- Branch unconditionally to address A
BSA A	- Branch to A and save return address
ISZ A	- Increment the value at A and skip if result is zero.

Input-output organization

Computer needs to communicate with the peripherals to take data or to transfer processed data.

Instructions and data stored must come from some Input device.

Similarly, the results of processing on data, must be

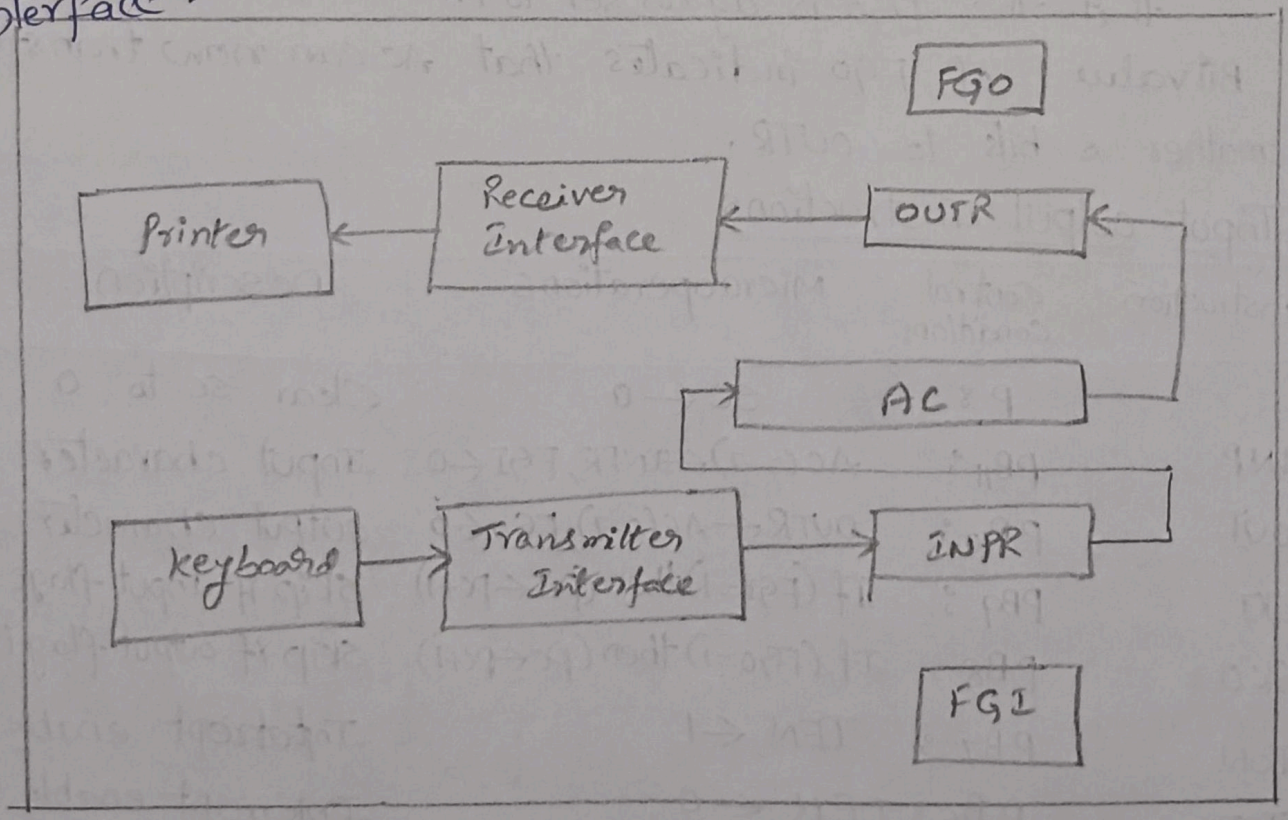
Subject: DECO
Faculty: A. Swathi
Topic: Input-output Instructions

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 9

sent to user through some output device.

- A commercial computer will be having multiple Input-output (I/O) devices, for simplicity, we will consider a keyboard as an input device and monitor (screen) as an output device.
- Keyboard is connected to the Input Register (INPR) through a transmitter interface. The register INPR accepts 8 bits from a keyboard at a time and sends them to AC.
- AC is connected to the OUTR, which sends 8 bits at a time to the monitor or printer through a receiver interface.



Input-output Configuration.

- Two single bit flipflops namely FGI and FGO control the transfer of data between memory and I/O devices.

→ When a new information is available in input device, then the single bit FGI flipflop is set to 1.
This information is transferred to AC.
After transfer has been done, FGI is set back to 0.
This indicates that INPR is now free and can accept another 8 bits of input data.
No data are transferred from keyboard to INPR as long as FGI contains the bit value 1.

- Another single bit flipflop FGO controls transmittion of data between Ac and OUTPR.

If FGO is set to 1, AC transfers data to OUTPR
if FGO is set to 0. it is only after transmission is completed, that the FGO is again set to 1.
Bitvalue 1 of FGO indicates that AC can now transfer another 8 bits to OUTR.

Input-output Instructions

Instruction	Control condition	Microoperations	Description.
	P :	SC ← 0	Clear sc to 0
INP	PB_{11} :	AC(0-7) ← INPR, FGI ← 0	Input character
OUT	PB_{10} :	OUTR ← AC(0-7), FGO ← 0	output character
SKI	PB_9 :	If (FGI=1) then (PC ← PC+1)	Skip if Input flage is on
SKO	PB_8 :	If (FGO=1) then (PC ← PC+1)	Skip if output flag is on
ION	PB_7 :	IEN ← 1	Interrupt enable on
IOF	PB_6 :	IEN ← 0	Interrupt enable OFF

Subject: DECO
Faculty: A. Swathi
Topic: Interrupt cycle

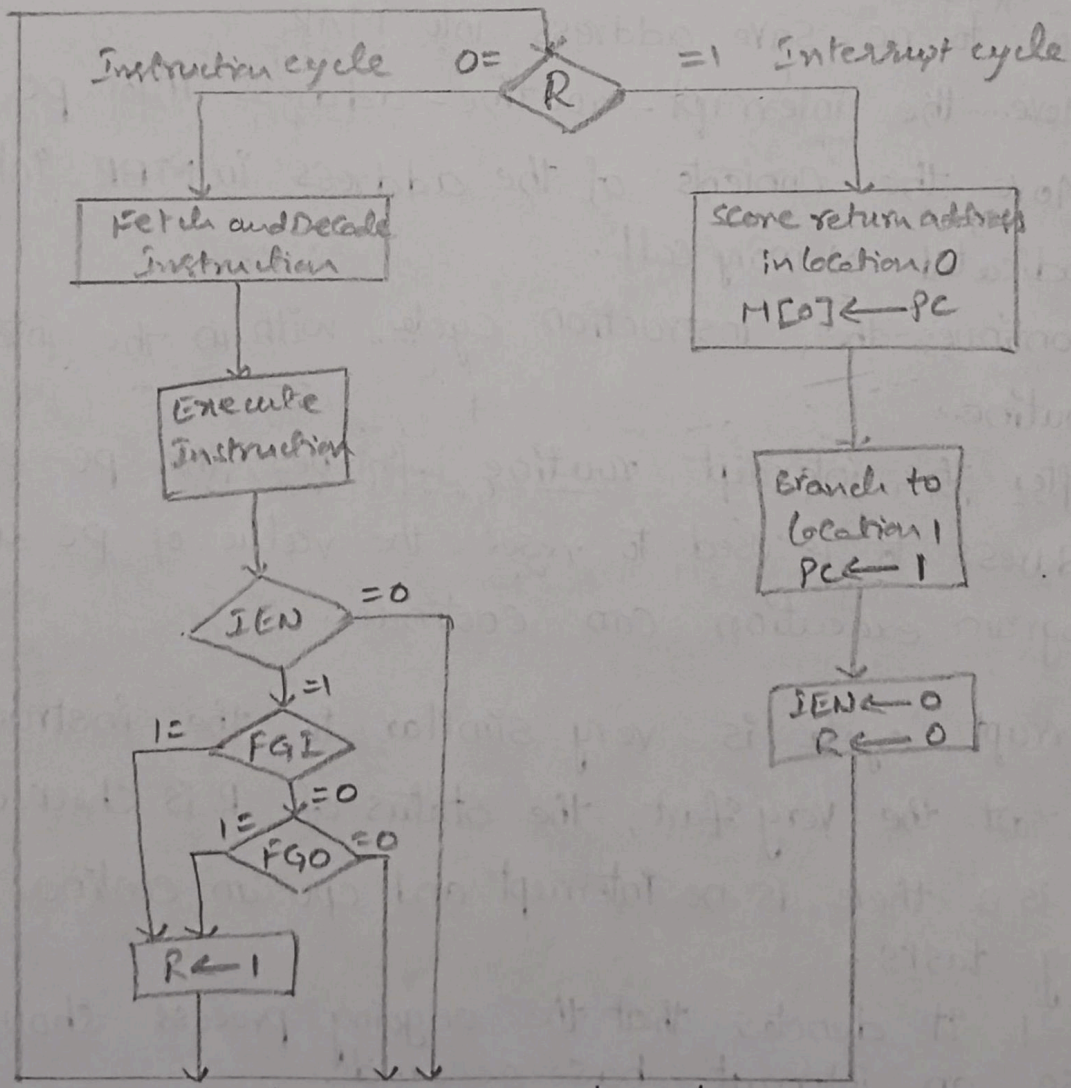
Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 10

Interrupt cycle

It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction requires, and carries out those actions.

This cycle is repeated continuously by the central processing unit (CPU), from bootup to when the computer is shutdown.



Flowchart for Interrupt cycle.

- When IEN it checks whether "FGI" or "FGO" are set to 1 then an interrupt cycle happens, but when FGI=1 it means that information in INPR cannot be changed, and FGO is the reverse to that which means that when FGO is set to 1 then the information in AC will be transferred to OUTR, OUTR can be changed.
- After the execute cycle is completed, a test is made to determine if an interrupt was enabled
  (ex: so that another process can access the CPU)
- If not, instruction cycle returns to the fetch cycle.
- If so, the interrupt cycle might performs the following tasks.
  - move to the current value of pc into MBR
  - Move to pc - save address into MAR.
  - Move the interrupt - routine - address into pc.
  - Move the contents of the address in MBR into indicated Memory cell.
  - Continue the instruction cycle with in the interrupt routine.
  - after the interrupt routine finishes, the pc - save address is used to reset the value of pc and program execution can continue.

→ Interrupt cycle is very similar to the instruction cycle. At the very start, the status of R is checked.
If it is 0 there is no interrupt and cpu can continue it's ongoing tasks.
But R=1, it denotes that the ongoing process should halt because an interrupt has occured.

Subject: DECO
Faculty: A. Swathi
Topic: Stack organization

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No ... of SP
Book Reference:
Date Conducted:
Page No: 11

Stack organization

A stack is a data storage structure in which the most recent thing deposited is the most recent item retrieved. It is based on the LIFO (Last In First out) concept.

- The stack is a collection of memory locations containing a register that stores the top of element address in digital computers.

stack's operations are :

push : Adds an item to the top of the stack.

pop : Removes one item from the stack's top.

LIFO The Last In First out list is another name for stack.

- It is the cpu's most crucial feature
- It saves information so that the last element saved is retrieved first.
- A memory space with an address register is called a stack. This register known as the stack pointer, affects the stack's address (SP).
- The Address of the element at the top of the stack is continuously influenced by the stack pointer.

Implementation of stack

The stack can be implemented using two ways.

- Register stack
- Memory stack.

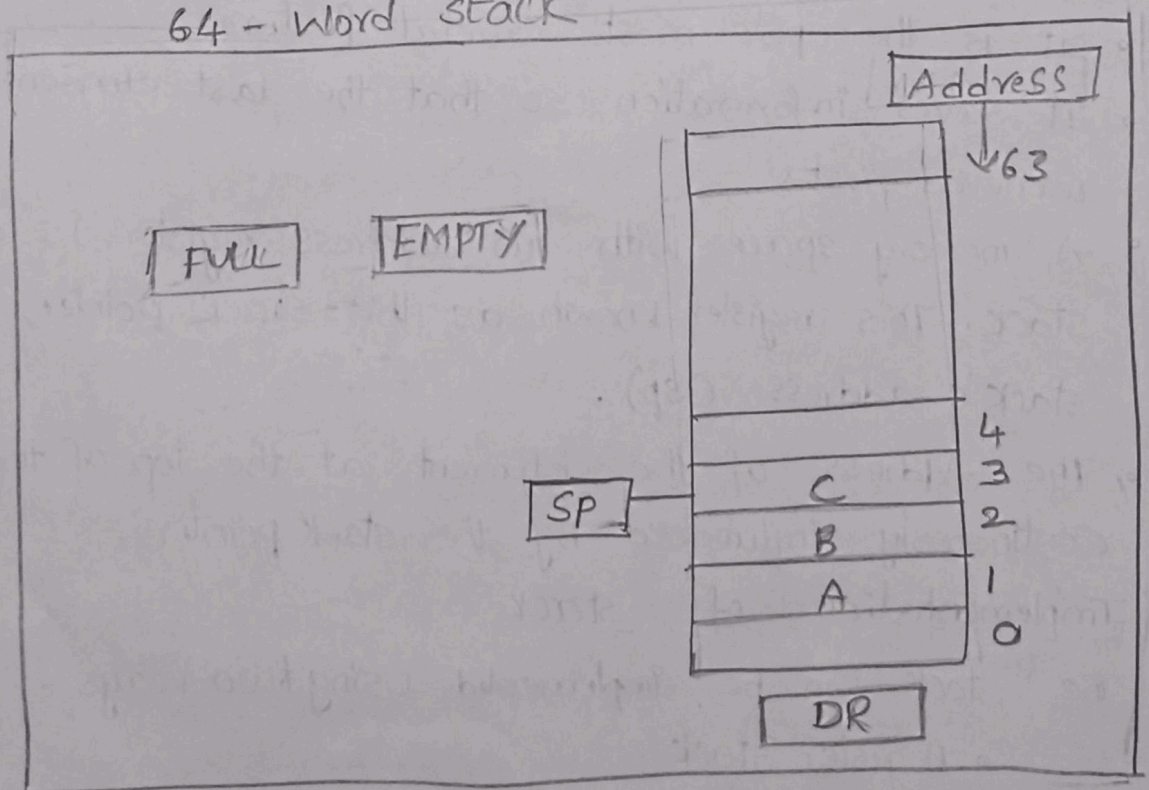
Register stack :

A stack of memory words or registers may be placed on top of each other.

Consider a 64-word register stack, A binary number which is the address of the element at the top of the stack, stored in the stack pointer register.

- The stack has the three elements A, B and C.
- The stack pointer holds C's address, which is 3.
  C is at the top of the stack.
  The top element is removed from the stack by reading the memory word at address 3 and decreasing the stack pointer by one.
- As a result, B is at the top of the stack, and the SP is aware of B's address, which is 2.

It may add a new word to the stack by increasing the stack pointer and inserting a word in the newly increased location.



Subject: DECO
Faculty: A. Swathi
Topic: stack organization

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 12

When a stack is created the pointer will point to the address 0 and the stack will be empty and not full.
Accordingly, the registers SP, EMPTY and FULL will be hold 0, 1 and 0 values respectively.

- When the FULL Register holds '0', new elements can be inserted into the stack using the push operation that involves the following oper sequence of microoperations.

```
SP ← SP+1              : stack pointer is incremented
M[SP] ← DR             : Element is inserted on top of the stack
If [SP=0] then (FULL ← 1) : Check whether the stack is full or not
EMPTY ← 0              : Indicate stack is not empty
```

- The top element of the stack can be deleted by performing pop operation only when the stack is not empty. The sequence of microoperations to pop the topmost element is listed below.

```
DR ← M[SP]             : Topmost element of stack is read in DR
SP ← SP-1              : stack pointer is decremented
If (SP=0) then (EMPTY ← 1) : Check whether the stack is empty
FULL ← 0               : To indicate that stack is not full
```

Memory stack

Stack operation can also be implemented in computer memories.

This can be done by assigning few segments of the main memory to store the stack, data and program instructions.

- Then the registers such as stack pointer (SP), Array Register (AR) and program counter (PC) will hold the address of topmost element of the stack, data and the instructions of the program, respectively.
- All these registers are connected to a common bus system. An additional Data Register (DR) is also used which help in communicating with the stack.

TO PUSH new element at the top of the stack, the following micro operations are performed.

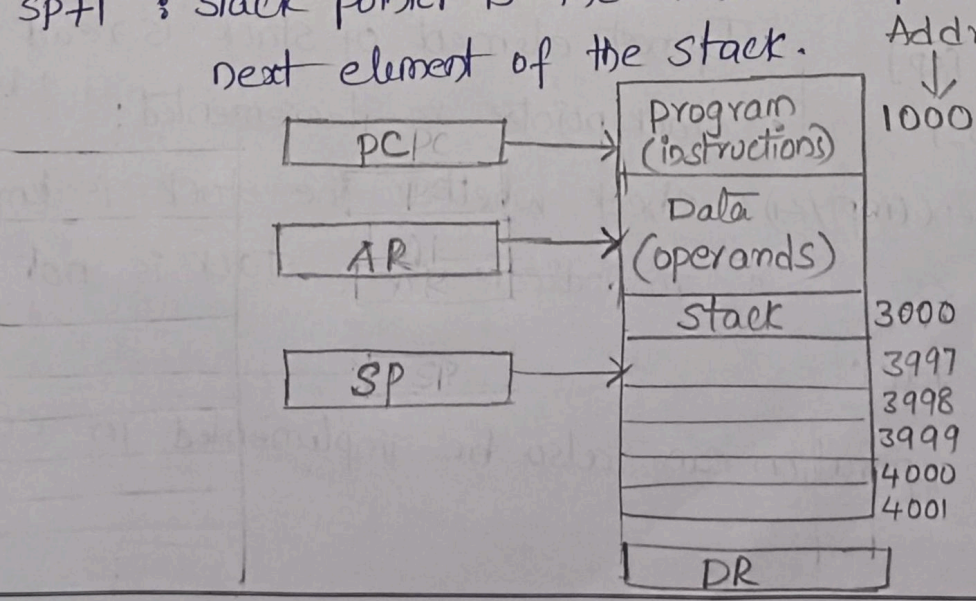
SP ← SP-1 : stack pointer is decremented to point to the newly inserted element

M[SP] ← DR : The element present in DR is inserted at top of stack

TO POP the topmost element of the stack, the following micro operations are performed.

DR ← M[SP] : The topmost element of stack stored in DR

SP ← SP+1 : stack pointer is incremented to point to the next element of the stack.



Subject: DECO
Faculty: A.Swathi
Topic: Reverse polish Notation in stack

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 13

Polish Notation:

Generally, the arithmetic expressions are written in infix notation as,

P * Q − R * S

In this notation, each operator is placed in between the operands.

The drawbacks while using this notation is that, it requires scanning of expression back and forth in order to find the next operation to be performed.

- The polish notation is actually the prefix notation and can be used to overcome the drawback of the infix notation,

In this notation the operator is placed before the operands

− * P Q * R S

Reverse polish Notation in stack

The reverse polish notation in the stack is also known as postfix expression.

Here, we use stack to solve the postfix expression.

- From the postfix expression, when some operand is found, we push it into the stack, and when some operator is found, we pop elements from the stack, and afterthat, the operation is performed in the correct sequence, and the result is also stored in the stack.

For example,

Given arithmetic expression is P*Q-R*S.

postfix notation for this is PQ*RS*-

post-fix notation can be used in place of infix notation to overcome its drawback.

Advantages of stack organization

- Complex arithmetic statements may be rapidly calculated.
- Instruction execution is rapid because operand data is stored in consecutive memory areas
- This instructions are minimal since they don't contain an address field.

Disadvantages of stack organization

- The size of the program increases when we use a stack
- It's in memory, and memory is slower in several ways than CPU registers.
  It generally has a lesser bandwidth and a longer latency.
  Memory accesses are more difficult to accelerate.

Subject: DECO
Faculty: A. Swathi
Topic: Addressing Modes

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 14

Addressing Modes:

Addressing Modes in computer Architecture plays a vital role in enabling efficient and flexible memory access and operand manipulation.

- Addressing modes define the rules and mechanisms by which the processor calculates the effective memory address or operand location for data operations.
- Each addressing mode has its own set of rules and mechanisms, allowing programmers to optimize memory utilization and enhance overall system performance.
- The different ways of specifying the location of an operand in an instruction are called addressing modes.

In Computer Architecture, there are following types of addressing modes -

1. Implied Addressing Mode
2. Immediate Addressing Mode
3. Direct Addressing Mode
4. Indirect Addressing Mode
5. Register direct Addressing Mode
6. Register Indirect Addressing Mode
7. Relative Addressing Mode
8. Indexed Addressing Mode
9. Base index Addressing Mode
10. Auto increment Addressing Mode
11. Auto decrement Addressing Mode.

Implied Addressing Mode:

In this addressing mode,
The definition of the instruction itself specify the operands implicitly.
It is also called as implicit addressing mode.

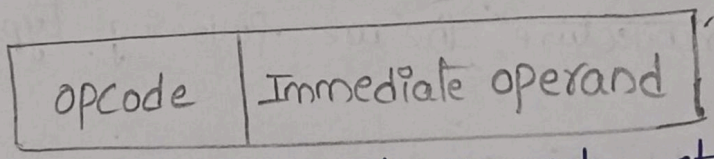
- As the operands are directly specified instead of their address.

Ex: Increment Accumulator.
The operand is the data present in the accumulator on which the increment operation is performed.

Immediate Addressing Mode:

In this addressing mode,
The operand is a part of the instruction instead of the contents of a register or memory location.

- The operand is specified in the instruction explicitly.



Ex: ADD 10 will increment the value stored in AC by 10.

- Immediate Addressing mode has an operand field rather than address field.
  since the data are encoded directly into the instruction, immediate operands normally represent constant data.

Direct Addressing Mode:

In this addressing mode,

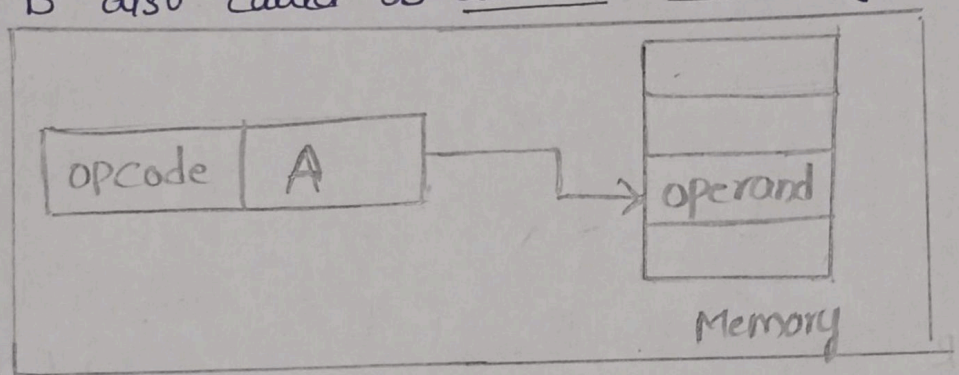
- The address field of the instruction contains the effective address of the operand.

Subject: OECO
Faculty: A.Swathi
Topic: Addressing modes

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S No. ... of SP
Book Reference:
Date Conducted:
Page No: 15

- only one reference to memory is required to fetch the operand.
- It is also called as absolute addressing mode.



Ex: ADD X

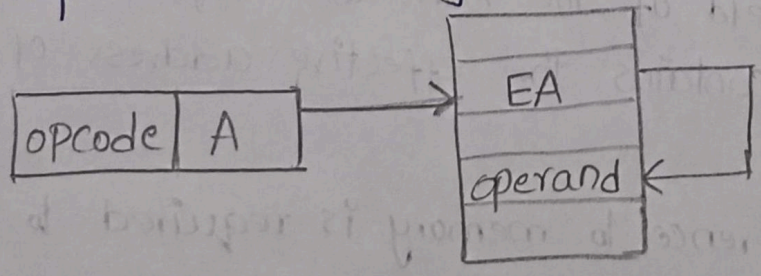
will increment the value stored in the Ac (accumulator) by the value stored at memory location X.

$$AC \leftarrow AC + [X]$$

Indirect Addressing Mode:

In this addressing mode,

- The address field of the instruction specifies the address of memory location that contains the effective address of the operand.
- Two references to memory are required to fetch the operand.



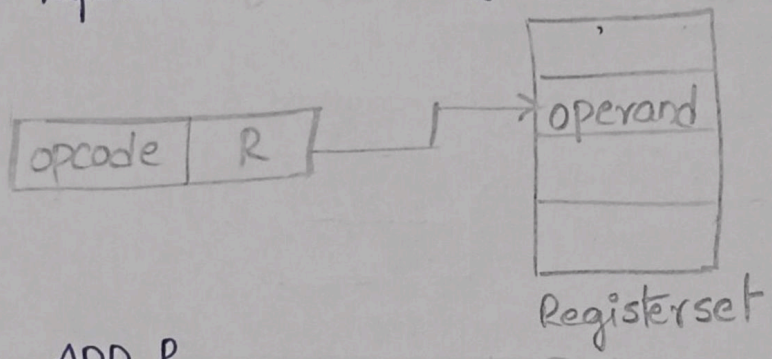
Ex: ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X

$$AC \leftarrow AC + [[X]]$$

Register Direct Addressing mode

In this addressing mode,

- The operand is contained in a registe set.
- The address field of the instruction refers to a CPU register that contains the operand.
- No reference to memory is required to fetch the operand.



Ex: ADD R

Will increment the value stored in the accumulator by the content of register R.

$AC \leftarrow AC + [R]$.

Note: Register Direct Addressing Mode is similar to Direct addressing mode.

The only difference is address field of the instruction refers to a cpu register instead of main memory

Register Indirect Addressing Mode –

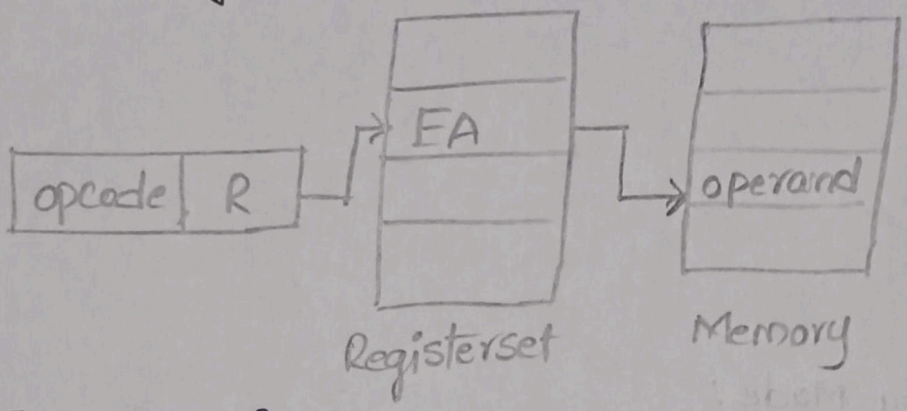
In this addressing mode,

- The address field of the instruction refers to a CPU register that contains the effective address of the operand.
- only one reference to memory is required to fetch the operand.

Subject: DECO
Faculty: A. Swathi
Topic: Addressing modes

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No .... of SP
Book Reference:
Date Conducted:
Page No: 16



Ex: ADD R

Will increment the value stored in the accumulator by the content of memory location specified in Register R

$$AC \leftarrow AC + [[R]]$$

NOTE: Register Indirect mode is similar to indirect addressing mode. The only difference is address field of the instruction refers to a cpu register.

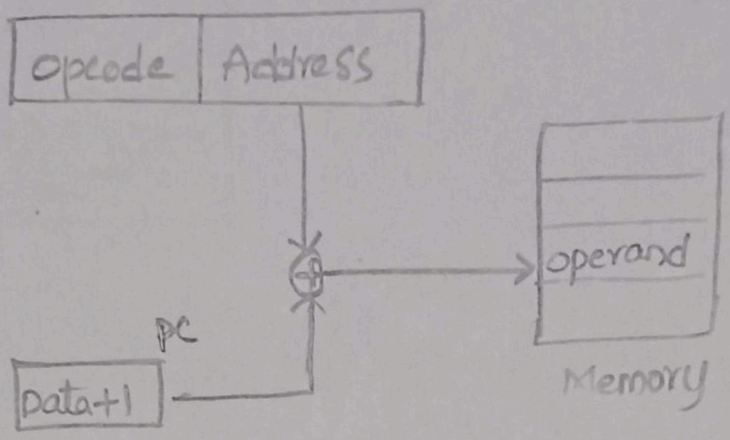
Relative Addressing mode:

In this addressing mode,

Effective address of the operand is obtained by adding the content of program counter with the address part of the instruction.

Effective Address = Content of program counter + Address part of the instruction

- Program Counter (PC) always contains the address of the next instruction to be executed.
- After fetching the address of instruction, the value of program counter immediately increases
- The value increases irrespective of whether the fetched instruction has completely executed or not.

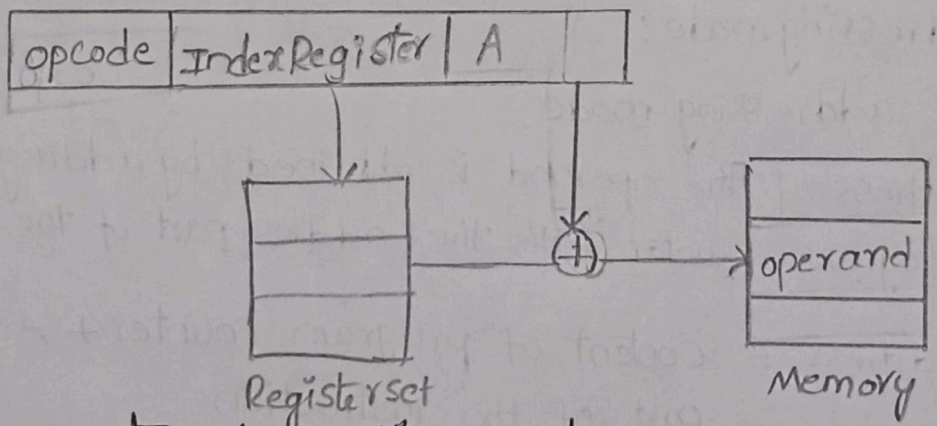


Indexed Addressing Mode:

In this addressing mode, the instruction contains an address.

- The address is added to the data present in the index register to get the effective address.

Effective address = Content of Index Register [IR] + Address specified in the instruction.



Base Register Addressing mode:

In this addressing mode, the instruction contains an address.

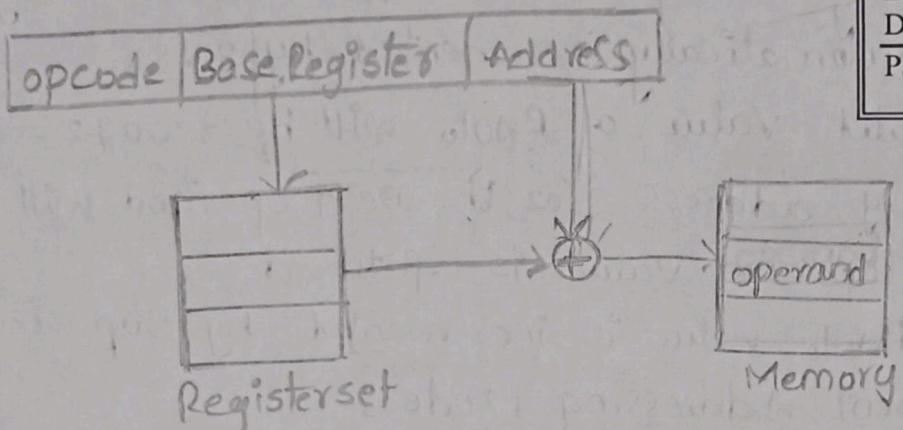
- This address is added to the data present in the index register to get the effective address

Effective address = Content of Base Register [BR] + Address specified in the instruction

Subject: DECO
Faculty: A. Swathi
Topic: Addressing modes

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No.... of SP
Book Reference:
Date Conducted:
Page No: 17



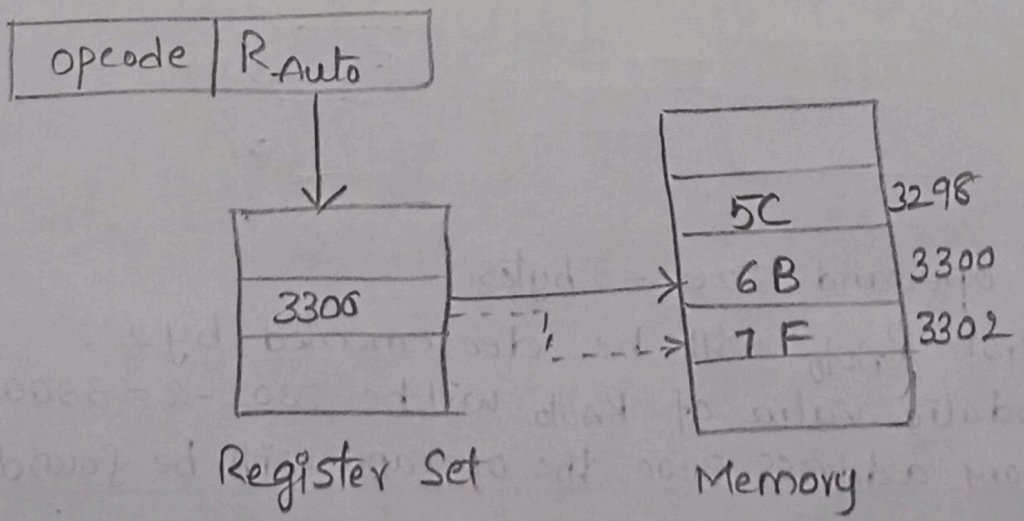
Auto Increment Addressing mode:

This addressing mode is a special case of Register indirect Addressing mode.

where
Effective Address of the operand = Content of Register

In this addressing mode,

- After accessing the operand, the content of the register is automatically incremented by step size 'd'.
- step size 'd' depends on the size of operand accessed.
- only one reference to memory is required to fetch the operand.



Assume operand size = 2 bytes

Here,

- After fetching the operand 6B, the instruction register R_{Auto} will be automatically incremented by 2.
- Then, updated value of R_{Auto} will be 3300+2 = 3302
- At memory address 3302, the next operand will be found →

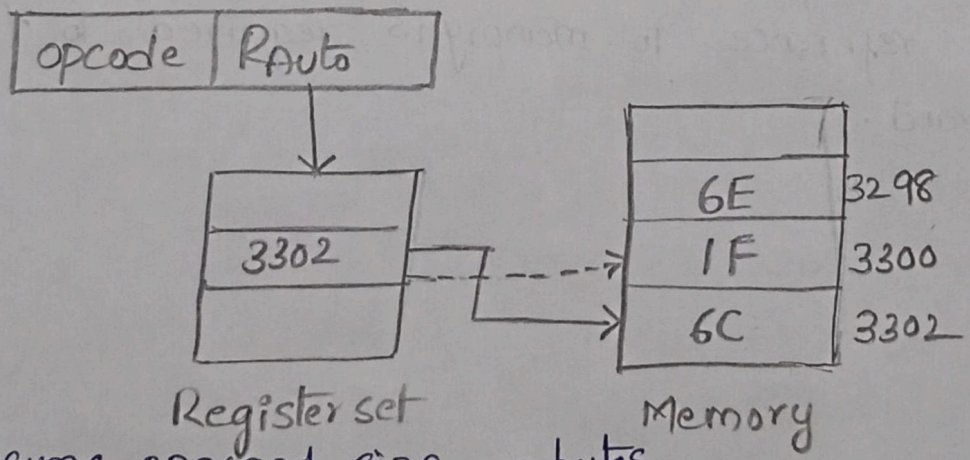
NOTE: 1) First operand value is fetched
2) Then R_{Auto} value is incremented by step size 'd'.

Auto-Decrement Addressing Mode:

In this addressing mode,

Effective Address of the operand = content of Register - stepsize.

- First, the content of the register is decremented by stepsize 'd'.
- Step size 'd' depends on the size of operand accessed.
- After decrementing, the operand is read.
- only one reference to memory is required to fetch the operand.



Assume operand size = 2 bytes

Here, First R_{Auto} will be decremented by 2.

Then, updated value of R_{Auto} will be 3302 - 2 = 3300

At Memory address 3300, the operand will be found.

Subject: DECo
Faculty: A-Swathi
Topic: Data transfer Instructions

Class Notes

Unit No: III
Lecture No:
Link to Session
Planner (SP): S.No ... of SP
Book Reference:
Date Conducted:
Page No: 18

Data transfer Instructions

Data transfer Instructions transfer the data between memory and processor registers, I/O devices and processor registers and from one processor register to another.

- There are 8 commonly used data transfer instructions.

Each instruction is represented by a mnemonic symbol.

Instruction Name	symbol used	Description
push	PUSH	It is used to direct a word of data on to the top of a stack.
pop	POP	It causes data word to be removed from the top of a stack
Input	IN	data to be loaded from a given source (I/O) into the memory or registers.
output	OUT	data to be transmitted from given memory or registers to the destination (I/O port, I/O devices etc.)
Load	LD	transfer of data from given memory location to the processor for further computations
Store	ST	reverse of load operation i.e data gets stored in memory
Move	MOV	transfer of data from source to destination.
Exchange	XCH	It causes swapping operation i.e the contents of source is copied into the destination and viceversa.

Data Manipulation Instructions:

Data Manipulation Instructions perform operations on data and provide computational capabilities for the computer.

- The data manipulation instructions in a typical computer are usually divided into three basic types.

1. Arithmetic instructions
2. Logical and bit manipulation instructions
3. Shift instructions

Arithmetic instructions:

As the name suggests arithmetic instructions cause arithmetic operations (add, subtract, division and multiplication etc) to be performed on a given set of data.

Instruction Name	Mnemonic Symbol	Description
Addition	ADD	It adds the values of two operands
Subtraction	SUB	It subtracts the values of two operands
Multiplication	MUL	It multiplies the two operands
Division	DIV	It divides the two operands by considering one as dividend & other as divisor and the result of this operation is the quotient.
Increment	INC	It increments the value of given operand by '1'
Decrement	DEC	It decrements the value of given operand by '1'.
Add with carry	ADDC	the addition operations b/n two operands and considers even its carry bit.
Subtract with	SUBB	The subtraction operation b/n two borrow operands and considers even the borrow bit.
Negation	NEG	It simply changes the sign of a given operand value.

Logical and Bit Manipulation Instructions

Logical instructions perform binary operations on strings of bits stored in registers. They are helpful for manipulating individual bits or a group of bits.

Instruction Name	mnemonic Symbol	Description
Logical AND	AND	It performs logical AND operation between the values of two registers.
Logical OR	OR	It performs logical OR operation between the values of two registers.
Logical exclusive OR	XOR	It performs logical ex-OR operation between the values of two registers.
Clear carry	CLRC	It clear or vacant the register maintaining the carry bit.
set carry	SETC	It enables the carry register
Clear	CLR	It empties or vacant the given register
Complement	COM	It inverts the bits of a given register
Enable Interrupt	EI	It enables an interrupt
Disable Interrupt	DI	It disables an interruct
Complement carry	COMC	It complements to inverts the carry bit.

Shift Instruction

Shift operations in which the bits of a word are moved to the left or right.
shift instructions may specify either logical shifts, arithmetic shifts or rotate type operations.

Instruction Name	Mnemonic symbol	Description
Logical shift left	SHL	shifted to 1 bit position towards its left excluding the Sign bit.
Logical shift right	SHR	shift right to be shifted 1 bit position towards its right excluding the sign bit.
Arithmetic shift right	SHRA	instructions to be shifted 1 bit position towards its right including the sign bit. But the sign bit remains unchanged
Arithmetic shift left	SHLA	shifted 1 bit ~~pos~~ towards its left, including the sign bit. But sign bit remains unchanged.
Rotate right	ROR	shifted 1 bit position towards its right in a circular manner i.e, the last bit of the register forming the 1st bit and rest of bits shifting towards their right by 1 bit position.
Rotate Left	ROL	shifted 1 bit position towards their left in a circular manner.
Rotate right through carry	RORC	same as ROR, except the fact that, it even carry includes the sign bit.
Rotate left Carry through	ROLC	Same as ROL, except the fact that, it even carry includes the sign bit.