FUNDAMENTALS OF COMPUTER ARCHITECTURE

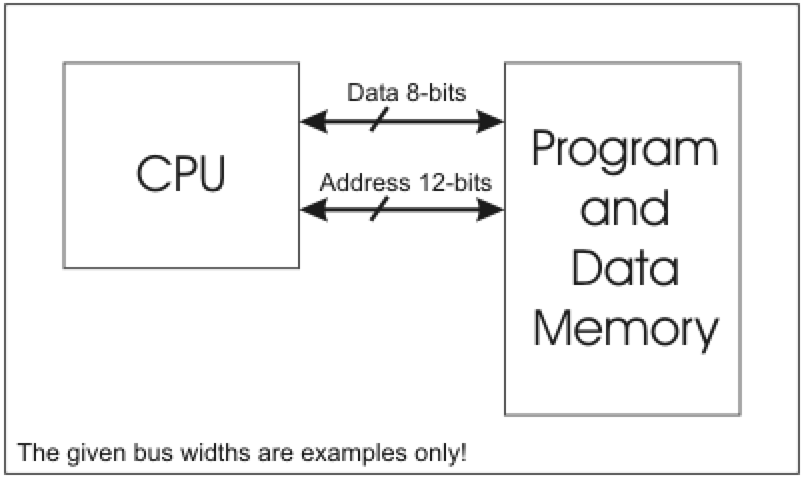
Unit - I

Topics of Unit - I

- Organization of the von Neumann machine
- Instruction formats
- The fetch/execute cycle, instruction decoding and execution;
- Registers and register files;
- Instruction types and addressing modes;
- Subroutine call and return mechanisms
- Programming in assembly language
- I/O techniques and interrupts
- Other design issues.

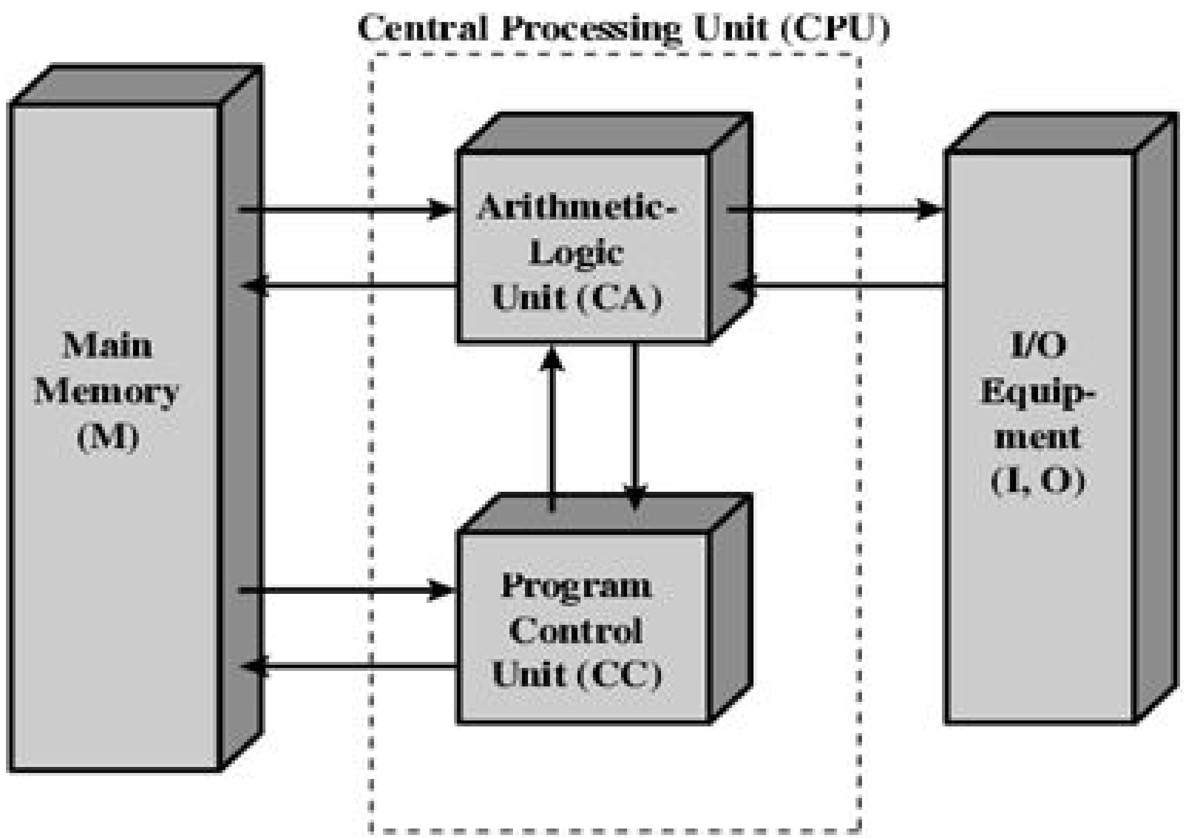
Von Neumann Architecture:

- Computer has ***single*** storage system(memory) for storing data as well as program to be executed.
- ***Processor needs two clock cycles*** to complete an instruction (Query and Reply)



- Pipelining is not possible
- This is a relatively older architecture and was replaced by Harvard architecture.

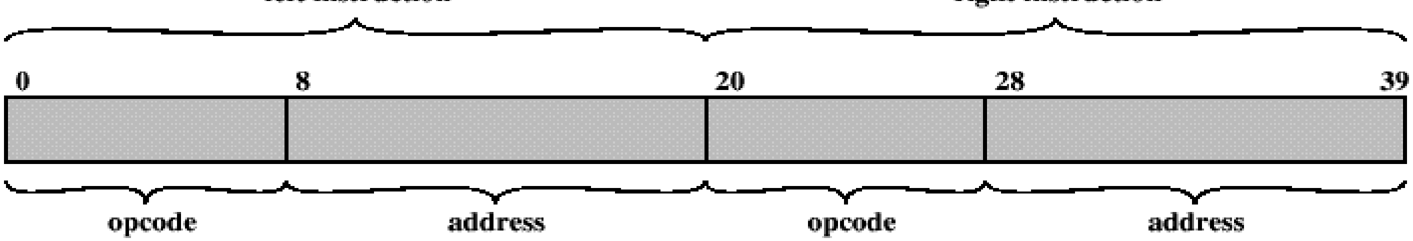
Structure of Von Neumann Machine



Memory of the IAS

- 1000 storage locations called words. (Institute for Advanced Studies (IAS)).
- Each word 40 bits.
- A word may contain:
  - A numbers stored as 40 binary digits (bits) – sign bit + 39 bit value
  - An instruction-pair. Each instruction:
    - An opcode (8 bits)
    - An address (12 bits) – designating one of the 1000 words in memory.

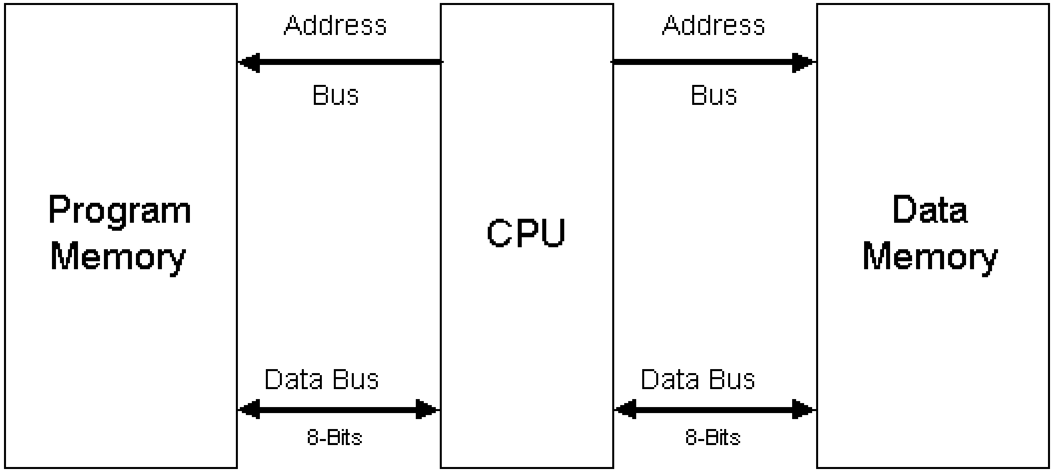
(a) Number word



(b) Instruction word

HARVARD ARCHITECTURE

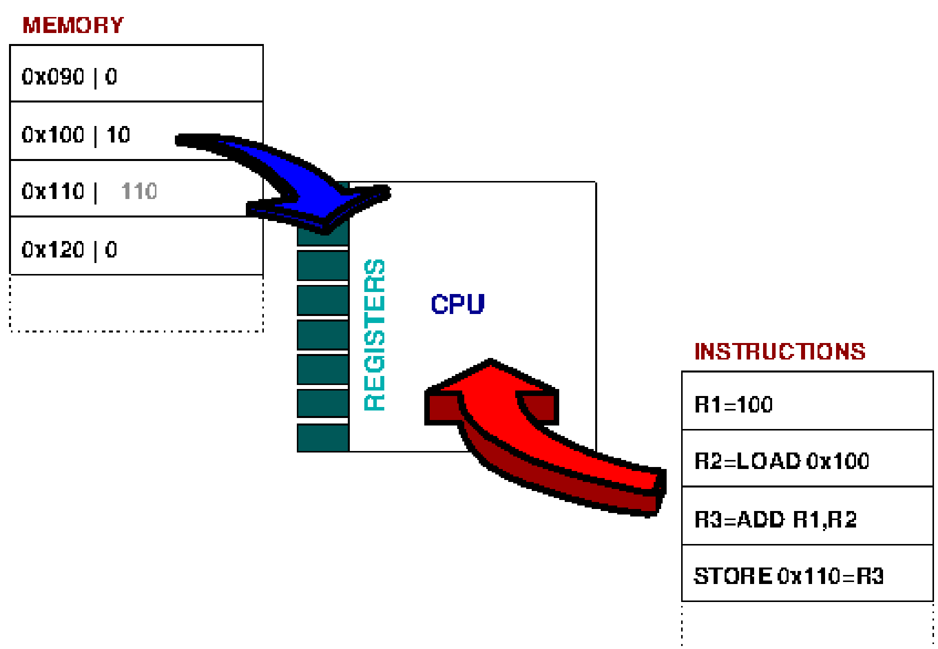
- Computer has two separate memories for storing data and program
- Processor can complete an instruction in one cycle
- Pipelining is possible



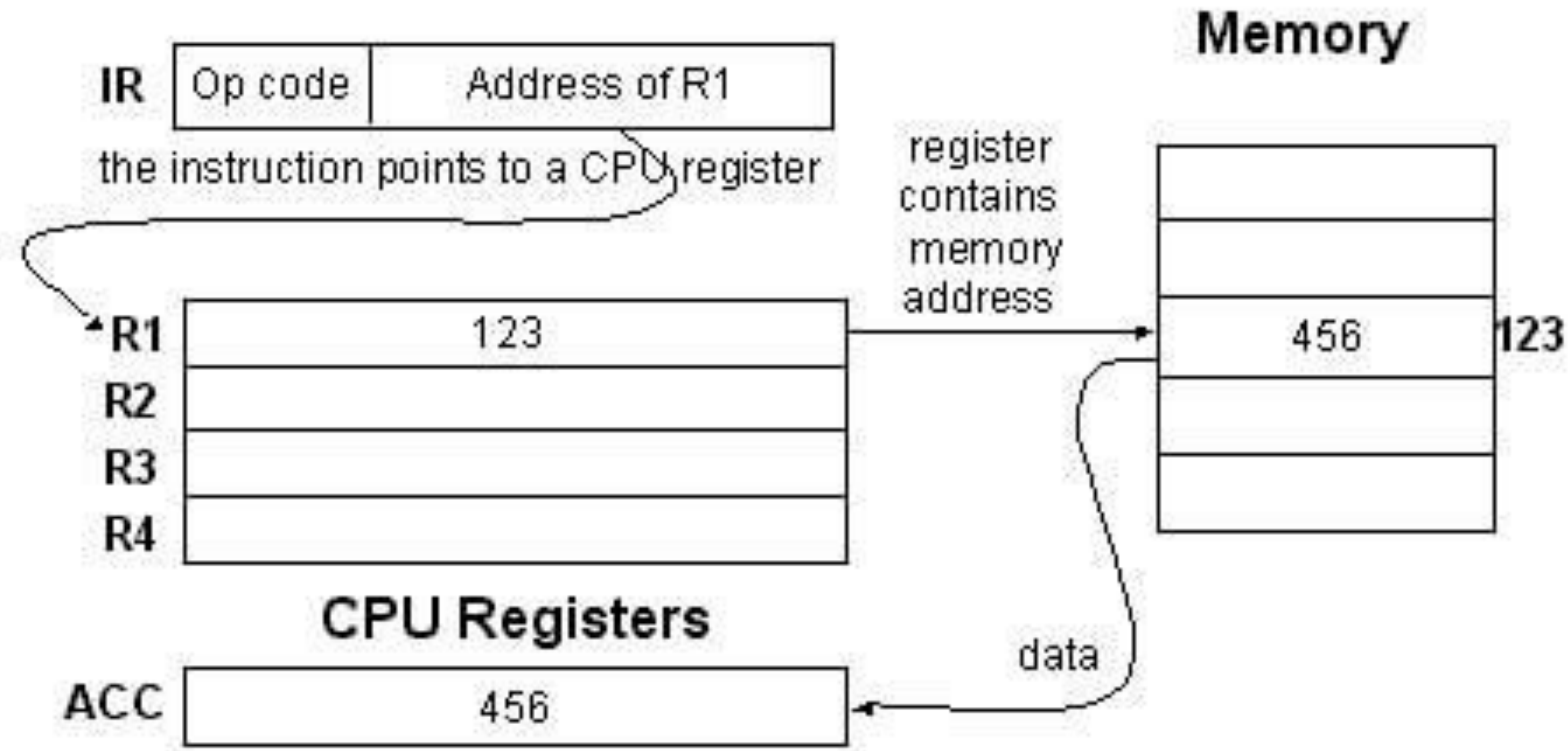
INSTRUCTION FORMAT

- An instruction set, or instruction set architecture (ISA), is the part of the computer architecture related to programming.

0 3	4 15
Opcode	Address

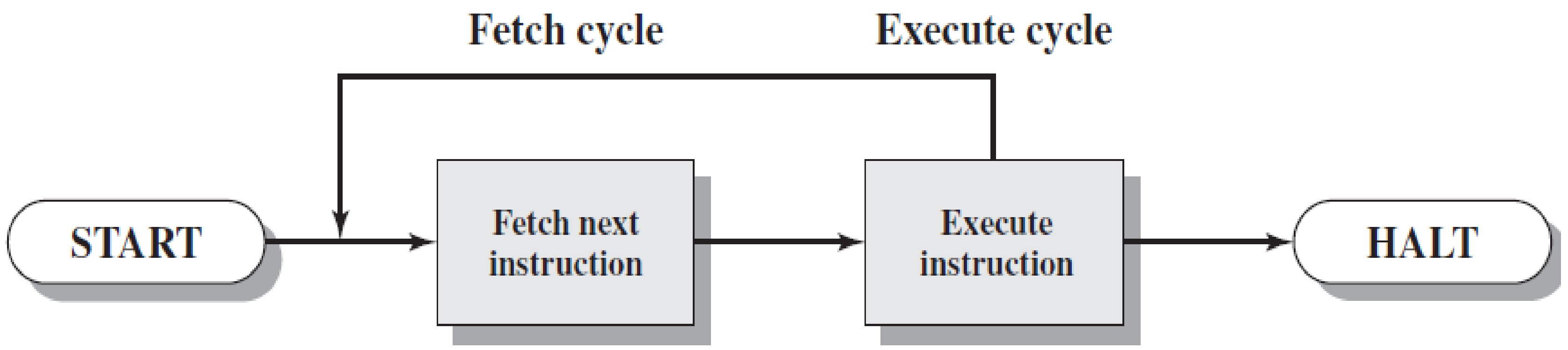


17-06-2021


IR
Op code
Address of R1
the instruction points to a CPU register
register contains memory address
Memory
R1
123
R2
R3
R4
456
123
CPU Registers
data
ACC
456

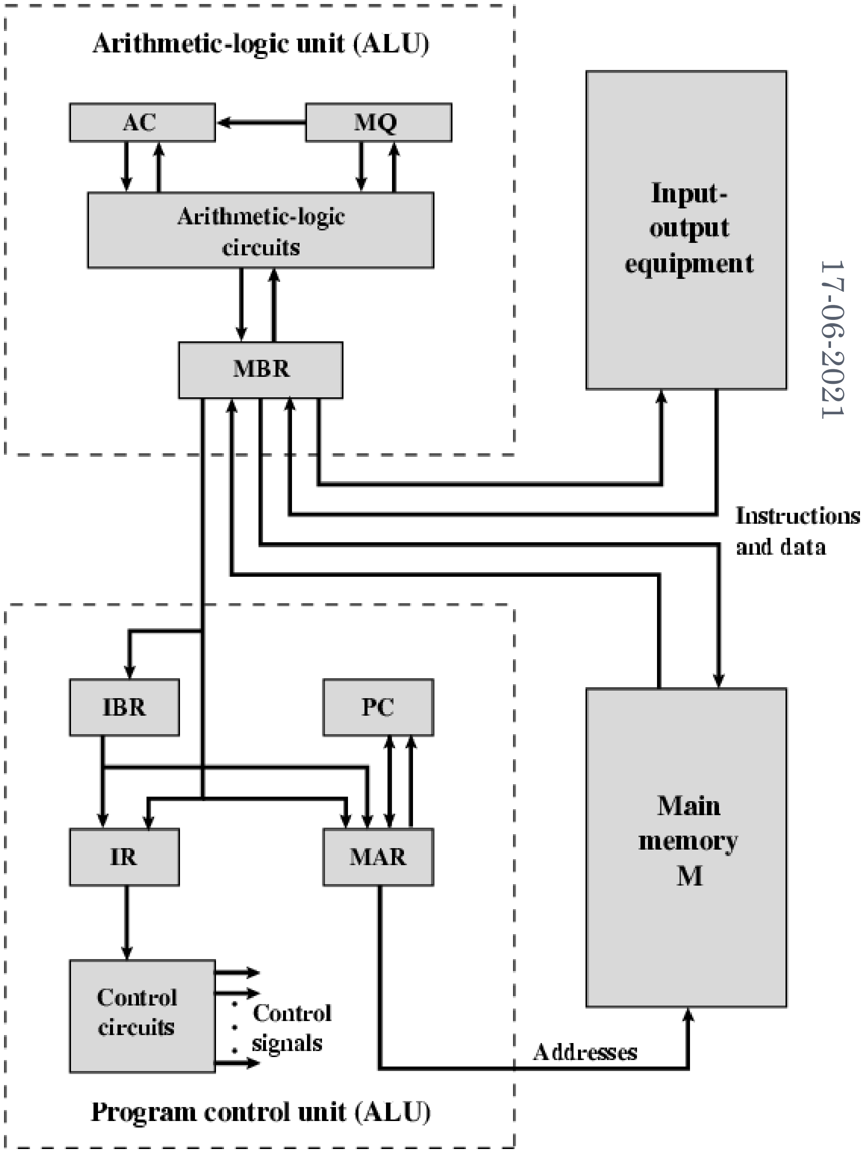
COMPUTER FUNCTION

- The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.
- Instruction fetch-decode-execute



17-06-2021

- **MBR: Memory Buffer Register** - contains the word to be stored in memory or just received from memory.
- **MAR: Memory Address Register** - specifies the address in memory of the word to be stored or retrieved.
- **IR: Instruction Register -** contains the 8-bit opcode currently being executed.
- **IBR: Instruction Buffer Register** - temporary store for RHS instruction from word in memory.
- **PC: Program Counter -** address of next instruction-pair to fetch from memory.
- **AC: Accumulator & MQ: Multiplier quotient** - holds operands and results of ALU ops.

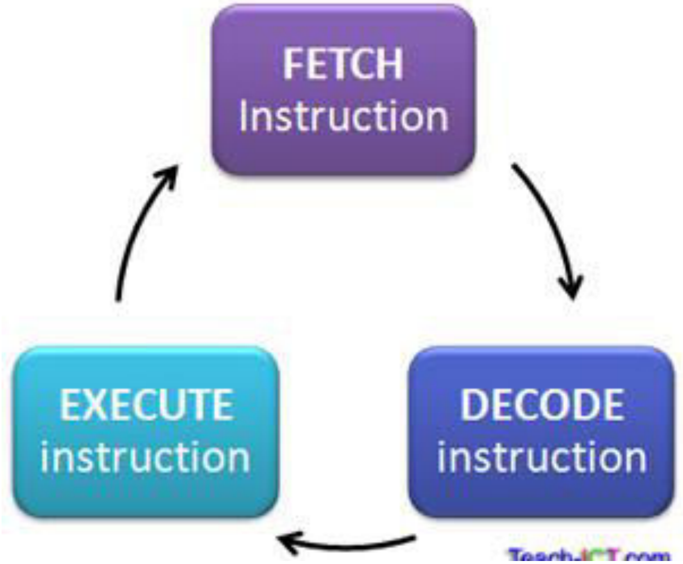


THE FETCH/EXECUTE CYCLE

- Standard process.

Also called as

- fetch-and-execute cycle,
- fetch-decode-execute cycle
- FDX



QUESTIONS:

- MBR –
- MAR –
- AC –
- IBR –
- IR –
- PC –
- MQ –
- IAS –
- What is Computer Architecture?
- What is Computer Organization?
- Number of words in IAS machine?
- Number of bits per word in IAS machine?
- Data is represented in ____________ form in IAS machine
- Explain Stored program concept.

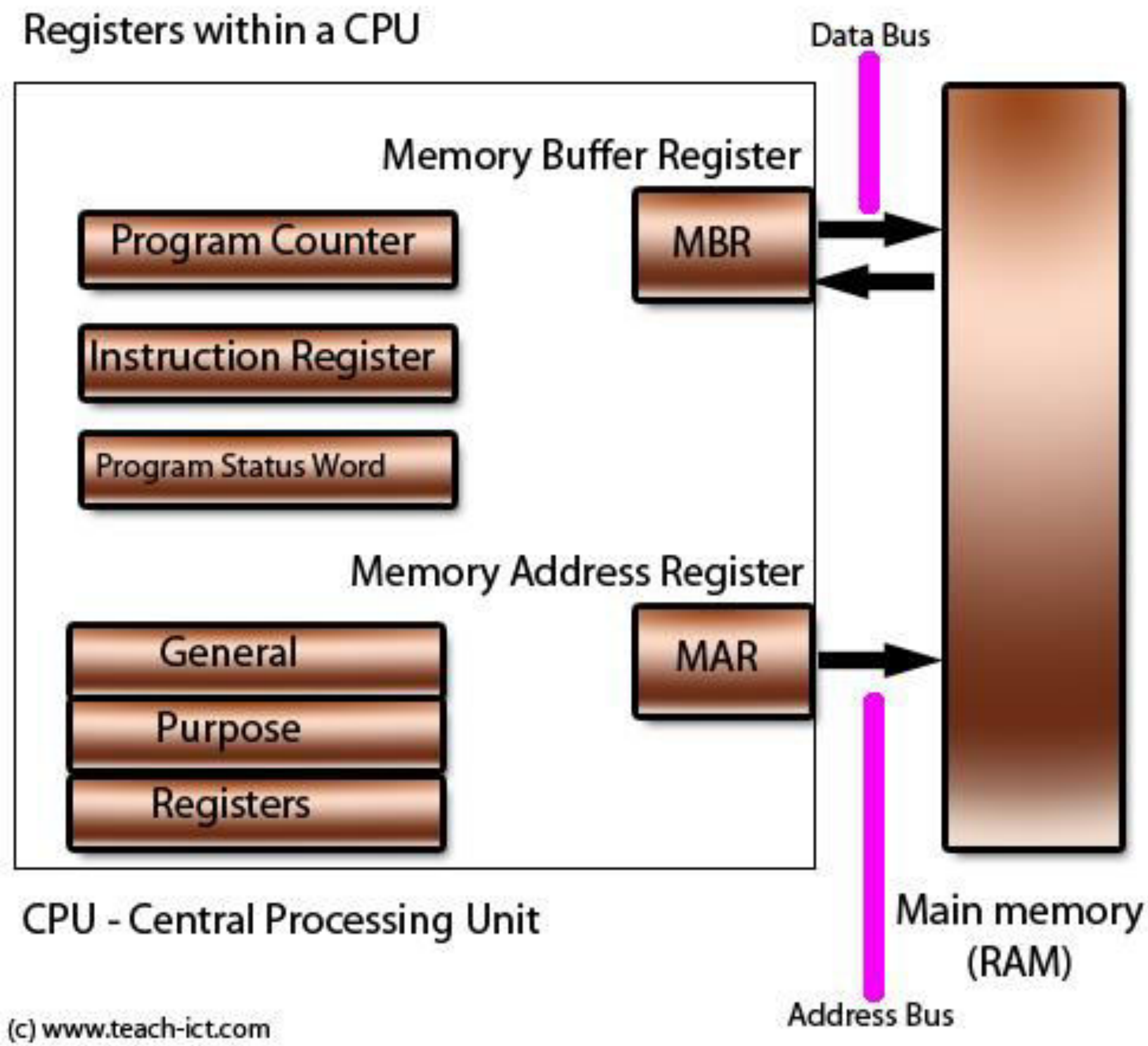
Registers and register files

- Registers?
  - Group of Flip-flops capable of storing one bit of information.
  - N bit registers consists of a group of N flip-flops capable of storing N bits.
  - Provides storage internal to the CPU.

As the instructions are interpreted and executed by the CPU, there is a movement of information between the various units of the computer system. In order to handle this process satisfactorily, and to speed up the rate of information transfer, the computer uses a number of special memory units, called registers. These registers are used to hold information on temporary basis, and are part of the CPU (not main memory).

CONT..

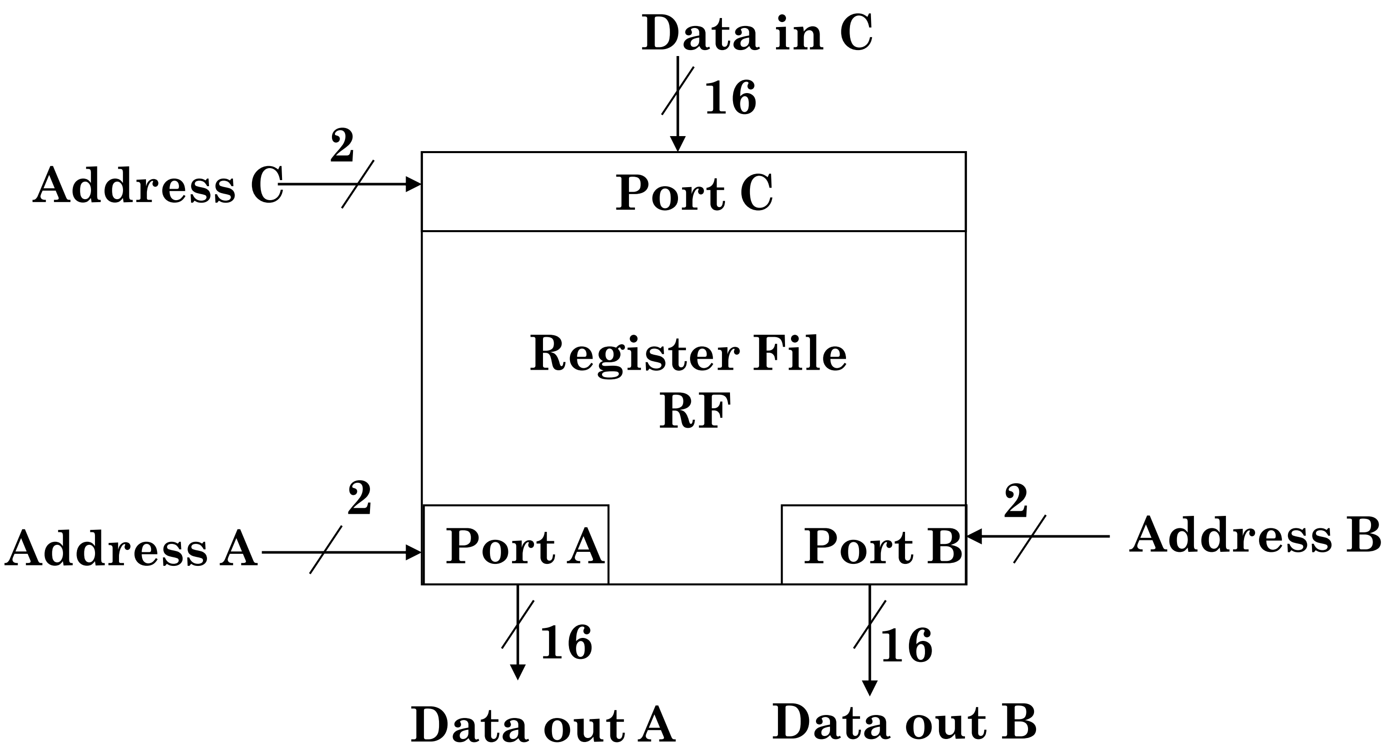
- The length of a register equals the number of bits it can store.
- Hence, a register that can store 8 bits is normally referred to as 8-bit register.
- Most CPU sold today, have 32-bit or 64-bit registers.
- The size of the registers is sometimes called the world size.
- The bigger the world size, the faster the computer can process a set of data.
- With all other parameters being same, a CPU with 32-bit registers, can process data twice as fast as one with 16-bit registers.


Registers within a CPU
Data Bus
Memory Buffer Register
Program Counter
MBR
Instruction Register
Program Status Word
Memory Address Register
General
Purpose
Registers
MAR
CPU - Central Processing Unit
Main memory
(RAM)
Address Bus
(c) www.teach-ict.com

Register Files (RF)

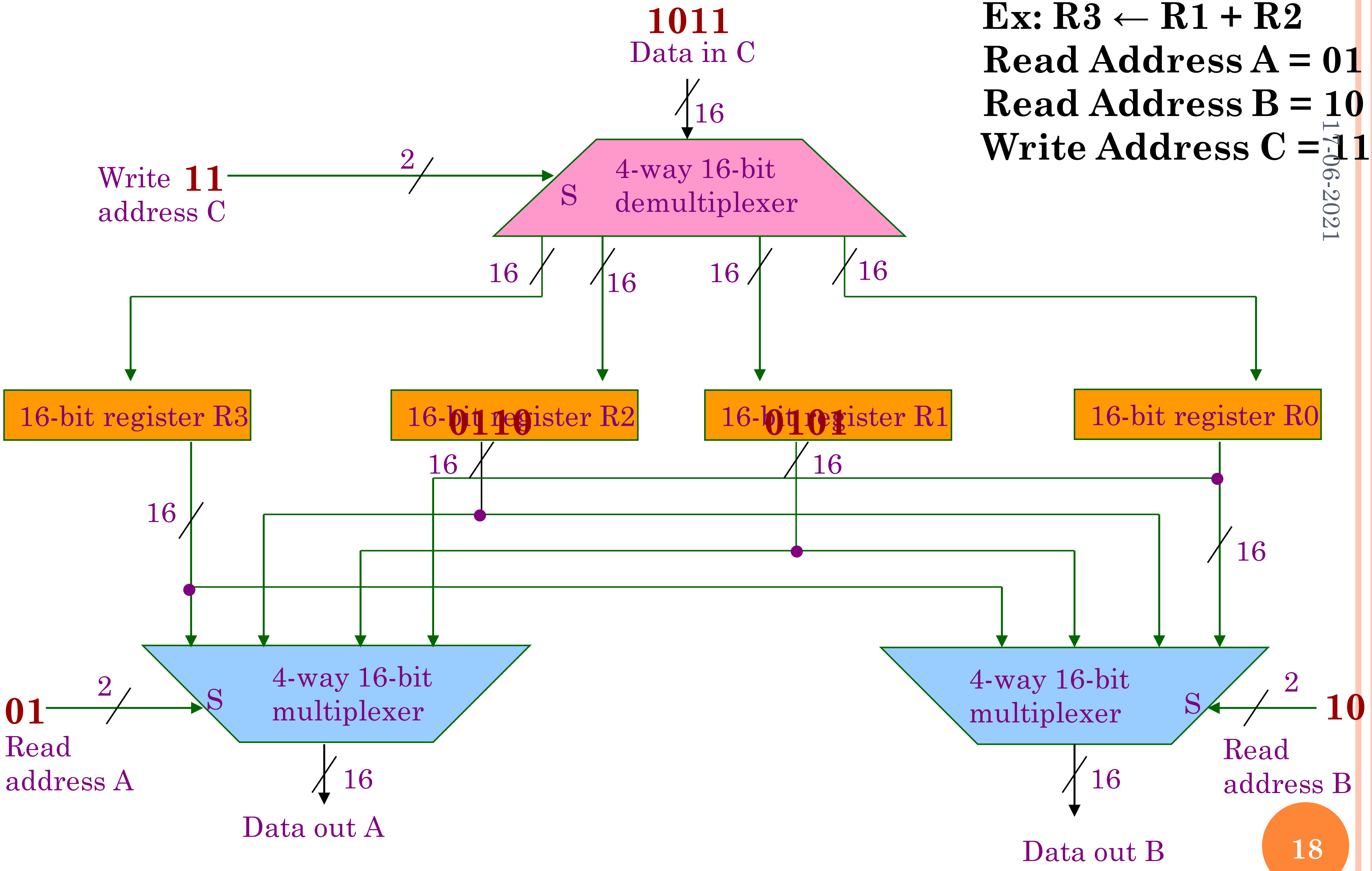
- Set of general purpose registers.
- It functions as small RAM and implemented using fast RAM technology.
- RF needs several access ports for simultaneously reading from or writing to several different registers. Hence RF is realized as multiport RAM.
- A standard RAM has just one access port with an associated address bus and data bus.

A REGISTER FILE WITH THREE ACCESS PORTS - SYMBOL



A Register File with three access ports – logic diagram

Ex: R3 ← R1 + R2
Read Address A = 01
Read Address B = 10
Write Address C = 11



17-06-2021

INSTRUCTION TYPES

- Data transfer instructions
- Data manipulation instructions
  - Arithmetic instructions
  - Logical and bit manipulation instructions
  - Shift instructions
- Program control instructions

Data transfer instructions

- Move data from one place to another without changing the data content in the computer.

17-06-2021

- Different data transfers:
  - Memory ↔ processor registers
  - Processor registers ↔ input or output
  - Processor register ↔ processor register

Set of data transfer instructions

- Load – transfer from memory to a processor register
- Store – transfer from processor register into memory
- Move – transfer from one register to another, transfer between register and memory or between two memory words.
- Exchange – swaps information between two registers or a register and a memory word
- Input – transfer data among registers/memory and input terminal
- Output – transfer data among register/memory and output terminal
- Push – transfer data from register/memory to memory stack
- Pop – transfer data from stack to register/memory

Data Manipulation Instructions

- Perform operations on data and provide the computational capabilities for the computer.
- Arithmetic instructions
  - Increment
  - Decrement
  - Add
  - Subtract
  - Multiply
  - Divide
  - Add with carry
  - Subtract with borrow
  - Negate (2's complement) – change the sign of the operand
  - Absolute – replace operand by its absolute value
  - Arithmetic shift left
  - Arithmetic shift right

Arithmetic Operations

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1	1
SUBB	A,#data	Subtract immed data from A with Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment data Pointer	1	2
MUL	AB	Multiply A and B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

Logic Operations

ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1

- Arithmetic shift left

0	0	0	1	1	0	1	0

Sign bit

0	0	0	1	1	0	1	0

Shift by 1 bit towards left

Sign bit

0

0	0	1	1	0	1	0	0

After shifting two times

Sign bit

0

0	1	1	0	1	0	0	0

After shifting three times

Overflow occurs as sign bit changes

Sign bit

0

Arithmetic shift Right

0	0	0	1	1	0	1	0

Sign bit

0	0	0	1	1	0	1	0

Sign bit

Shift by 1 bit towards right

0	0	0	0	1	1	0	1

Sign bit

After shifting two times

0	0	0	0	0	1	1	0

Sign bit

After shifting three times

17-06-2021

Data manipulation instructions

- Logical and Bit manipulation instructions
  - Clear (can also be included in data transfer instruction based on the way the operation is performed – 0's transferred to the destination)
  - Complement
  - AND- to clear a bit
  - OR – set a bit
  - Ex-Or –to complement a bit
  - Clear carry
  - Set carry
  - Complement carry
  - Enable interrupt – flip-flop that controls the interrupt facility is enabled
  - Disable interrupt – flip-flop that controls the interrupt facility is disabled

Data manipulation instructions

- Shift Instructions
  - Logical left shift
  - Logical right shift
  - Arithmetic shift left
  - Arithmetic shift right
  - Rotate right
  - Rotate left
  - Rotate right through carry
  - Rotate left through carry

- Logical shift left

0	0	0	1	1	0	1	0

0	0	0	1	1	0	1	0

Shift by 1 bit towards left

0

0	0	1	1	0	1	0	0

After shifting two times

0

0	1	1	0	1	0	0	0

After shifting three times

0

- Logical shift Right

	0	0	0	1	1	0	1	0	
0	0	0	0	1	1	0	1	0	**Shift by 1 bit towards right**
0	0	0	0	0	1	1	0	1	**After shifting two times**
0	0	0	0	0	0	1	1	0	**After shifting three times**

- Rotate left

0	0	0	1	1	0	1	0

Buffer

0	0	0	1	1	0	1	0

Rotate by 1 bit towards left

Buffer

0	0	1	1	0	1	0	0

After rotating two times

Buffer

0	1	1	0	1	0	0	0

After rotating three times

- Rotate right

0	0	0	1	1	0	1	0

rotate by 1 bit towards right

0	0	0	1	1	0	1	0

Buffer

0	0	0	0	1	1	0	1

After rotating two times

Buffer

1	0	0	0	0	1	1	0

After rotating three times

Buffer

- Rotate left through carry

0	0	0	1	1	0	1	0

Rotate by 1 bit towards left

Buffer									Carry
	0	0	0	1	1	0	1	0	0

After rotating two times

Buffer									Carry
	0	0	1	1	0	1	0	0	0

After rotating three times

Buffer									Carry
	0	1	1	0	1	0	0	0	0

- Rotate right through carry

0	0	0	1	1	0	1	0

rotate by 1 bit towards right

Carry									Buffer
0	0	0	0	1	1	0	1	0	

After rotating two times

Carry									Buffer
0	0	0	0	0	1	1	0	1	

After rotating three times

Carry									Buffer
0	1	0	0	0	0	1	1	0	

Program control Instructions

- Branch
- Jump
- Skip
- Call
- Return
- Compare (by subtraction)
- Test (by ANDing)

Addressing Modes

The way the operands are chosen during the program execution is dependent on the addressing mode of the instruction.

Different Types

- Implied Addressing Mode
- Immediate Addressing Mode
- Direct Addressing Mode
- Indirect Addressing Mode
- Register Direct Addressing Mode
- Register Indirect Addressing Mode
- Displacement Addressing Mode (combines the direct addressing and register addressing modes)
  - Relative Addressing Mode
  - Indexed Addressing Mode
  - Base Addressing Mode
- Auto Increment and Auto Decrement Addressing Mode

Implied Addressing Mode

- No address field is required
- Operand is implied / implicit
- Ex:
  - Complementing Accumulator
  - Set or Clearing the flag bits (CLC, STC etc.)
- 0 – address instructions in a stack organized computer are implied mode instructions.
- Effective Address (EA) = AC or Stack[SP]
- Ex: Tomorrow, I am on leave (implies that there is no CAO class)
- Come to my cabin (implies to come to 313A-07 SJT)

Immediate Addressing Mode

- Operand is specified in the instruction itself
- Useful for initializing the registers with constant value
- Operand = address field
- Ex: Mov Dx, #0034H
- Advantage: No memory Reference, fast
- Disadvantage: Limited operand magnitude
- Ex: Come to my cabin: 313A-07 SJT

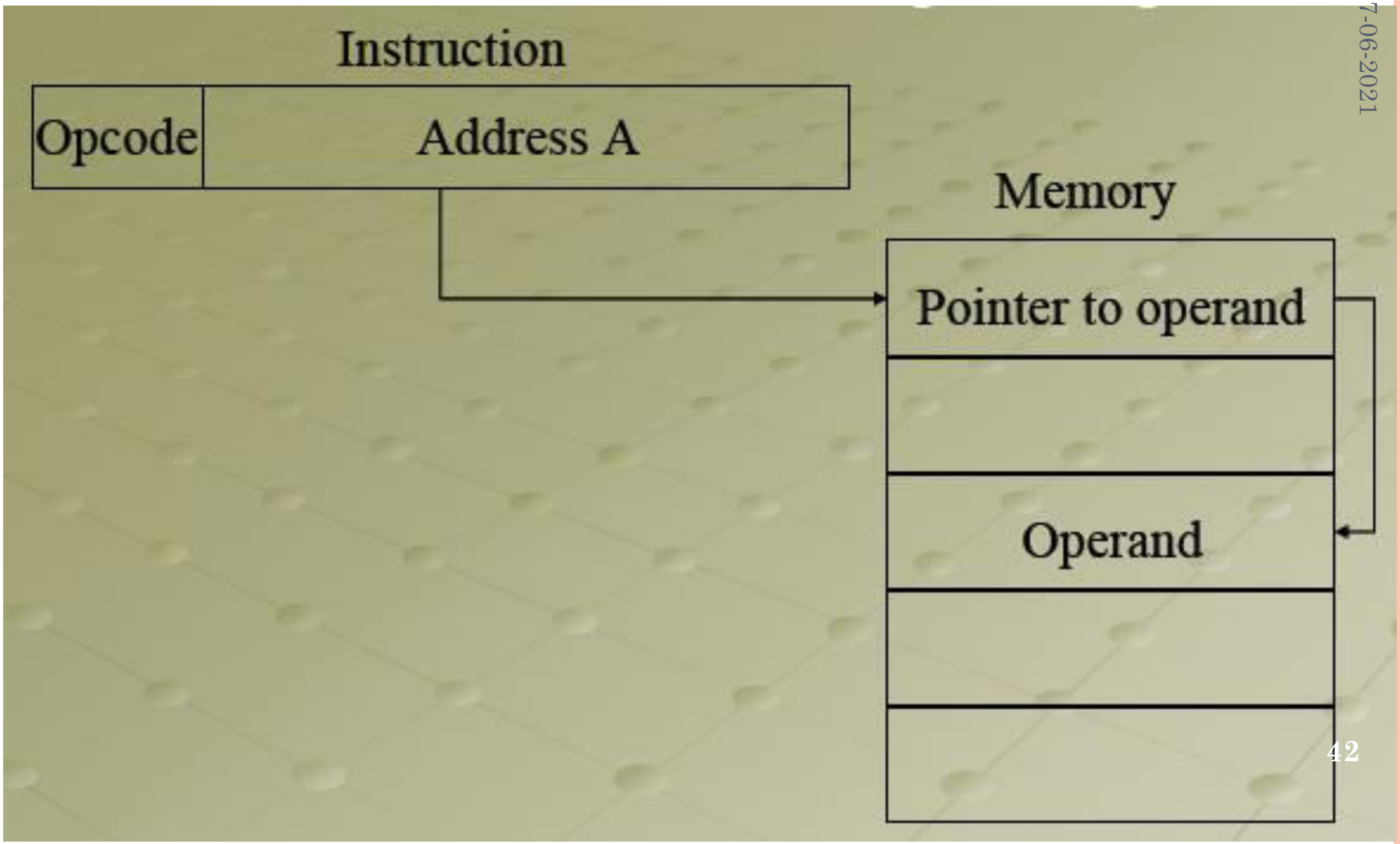
DIRECT ADDRESSING MODE

- Effective address is the address part of the instruction
- EA (effective address) = A
- Ex:
- Mov CX, [4200]H
- Advantage: Simple memory reference to access data, no additional calculations to work out effective address
- Disadvantage: Limited address space
- Ex: Aashiq, please bring my laptop from my cabin (cabin is known to Aashiq)

Indirect Addressing Mode

- The address field of the instruction gives the address of the effective address of the operand stored in the memory.
- EA = (A)
- Ex: Mov CX, [BX]
- Advantage: Large address space, may be nested, multilevel or cascaded
- Disadvantage: Multiple memory accesses to find the operand, hence slower

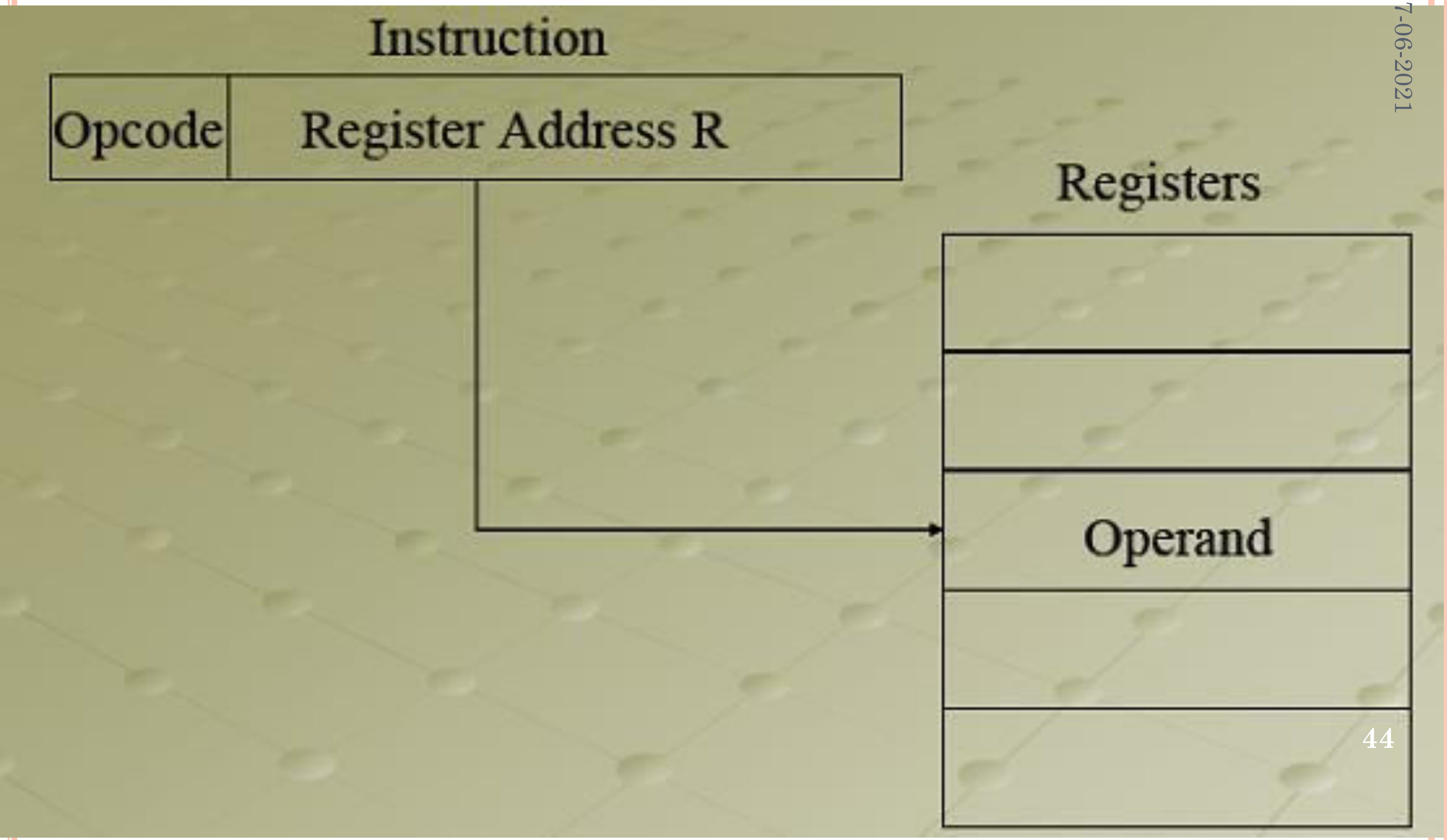
Indirect Addressing Mode Diagram



Register Direct Addressing Mode

- Operand is in the register specified in the address part of the instruction
- EA = R
- Ex: Mov AX, [BX]
- Special case of direct addressing
- Advantage: No memory reference, shorter instructions, faster instruction fetch, very fast execution
- Disadvantage: Limited address space as limited number of registers

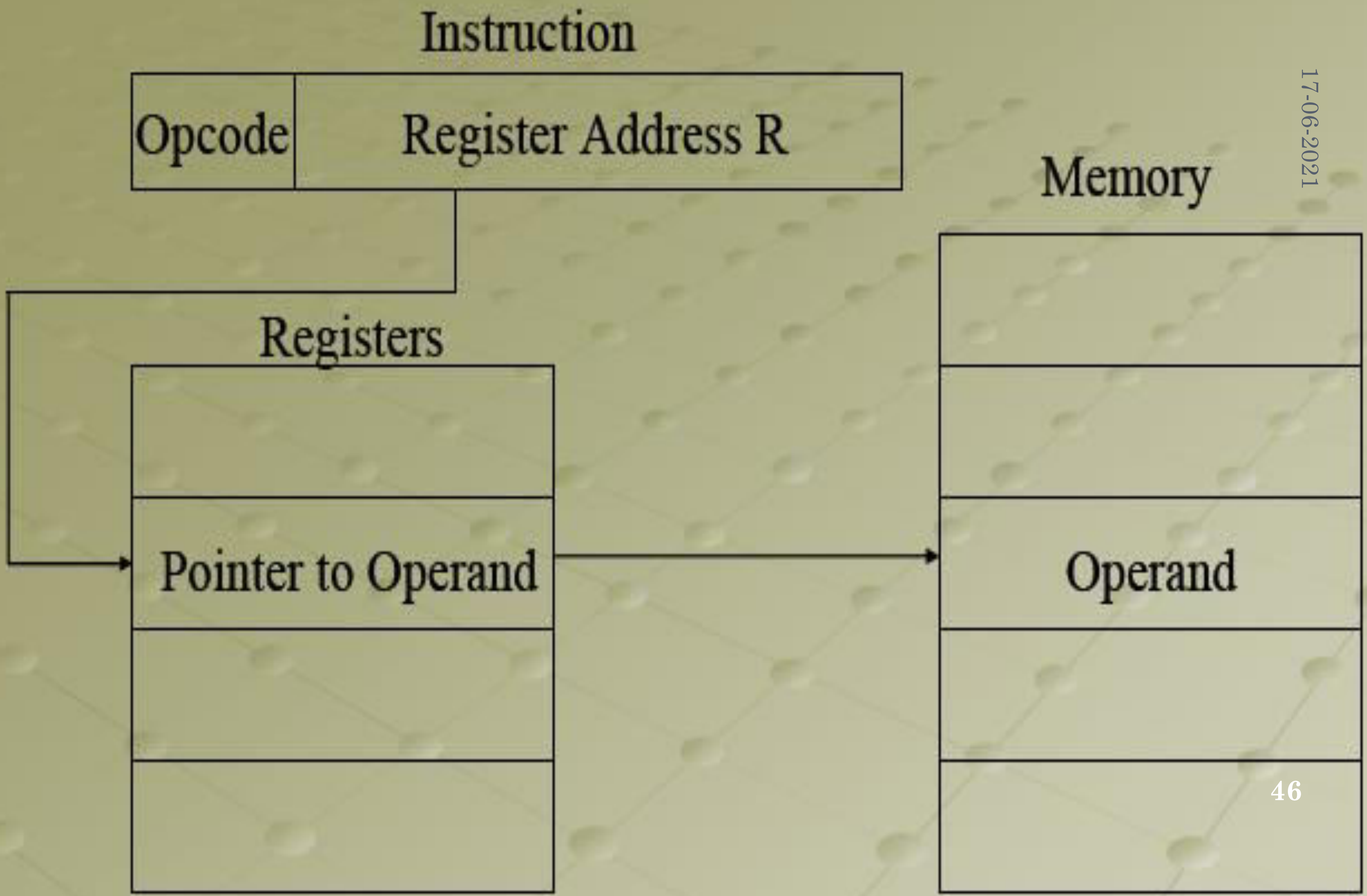
Register Addressing Diagram



Register Indirect Addressing Mode

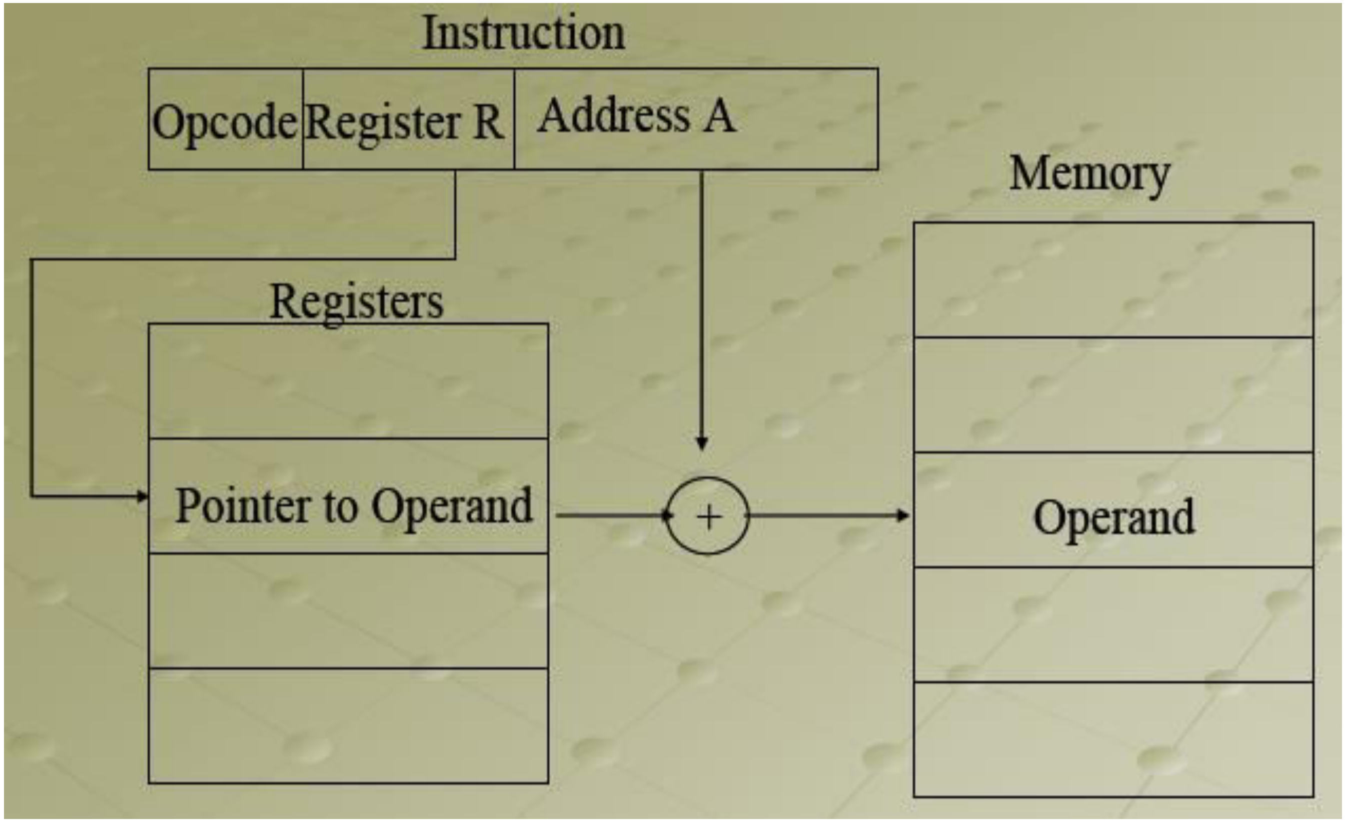
- Address part of the instruction specifies the register which gives the address of the operand in memory
- Special case of indirect addressing
- EA = (R)
- Ex: Mov BX, [DX]
- Advantage: Large address space
- Disadvantage: Extra memory reference

Register Indirect Addressing Diagram



Displacement Addressing Mode

- EA = A + (R)
- Address field holds two values
  - A = Base value
  - R = register that holds displacement
  - Or vice-versa



17-06-2021

RELATIVE ADDRESSING MODE

- Version of the displacement addressing
- R = program counter, PC
- Content of PC is added to address part of the instruction to obtain the effective address of the operand
- EA = A + (PC)
- It is often used in branch (conditional and unconditional) instructions, locality of reference and cache usage
- Advantage: Flexibility
- Disadvantage: Complexity

Indexed Addressing Mode

- A holds base address
- R holds displacement, may be explicit or implicit (segment registers in 8086)
- Content of the index register is added to the address part of the instruction to obtain effective address of the operand.
- Used in performing iterative operations
- EA = A + (SI)
- Ex: Mov CX, [SI] 2400H
- Advantage: Flexibility, good for accessing arrays
- Disadvantage: Complexity

Base Register Addressing Mode

- The content of the base register is added to the address part of the instruction to obtain the effective address of the operand.
- Used to facilitate the relocation of programs in memory.
- EA = A + (BX)
- Ex: Mov 2345H [BX], 0AC24H
- Advantage: Flexibility
- Disadvantage: Complexity

Auto Increment and Auto Decrement Addressing Modes

- This addressing mode is used when the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table.
- Ex: Mov AX, (BX)+, Mov AX, -(BX)
- Used mostly in Motorola 680X0 series of computers

Basic Addressing Modes differences :

	Mode	Algorithm	Advantage	Disadvantage
1	Immediate	Operand=1	No memory Reference	Limited operand magnitude
2.	Direct	EA = A	Simple	Limited address space
3	Indirect	EA =(A)	Large Address space	Multiple Memory References
4.	Register	EA = R	No memory Reference	Limited address space
5.	Register Indirect	EA = (R)	Large address space	Extra memory space
6	Displacement	EA = A+(R)	Flexibility	Complexity
7.	Stack	EA= Top of Stack	No memory Reference	Limited Applicability

16 BIT ADDITION

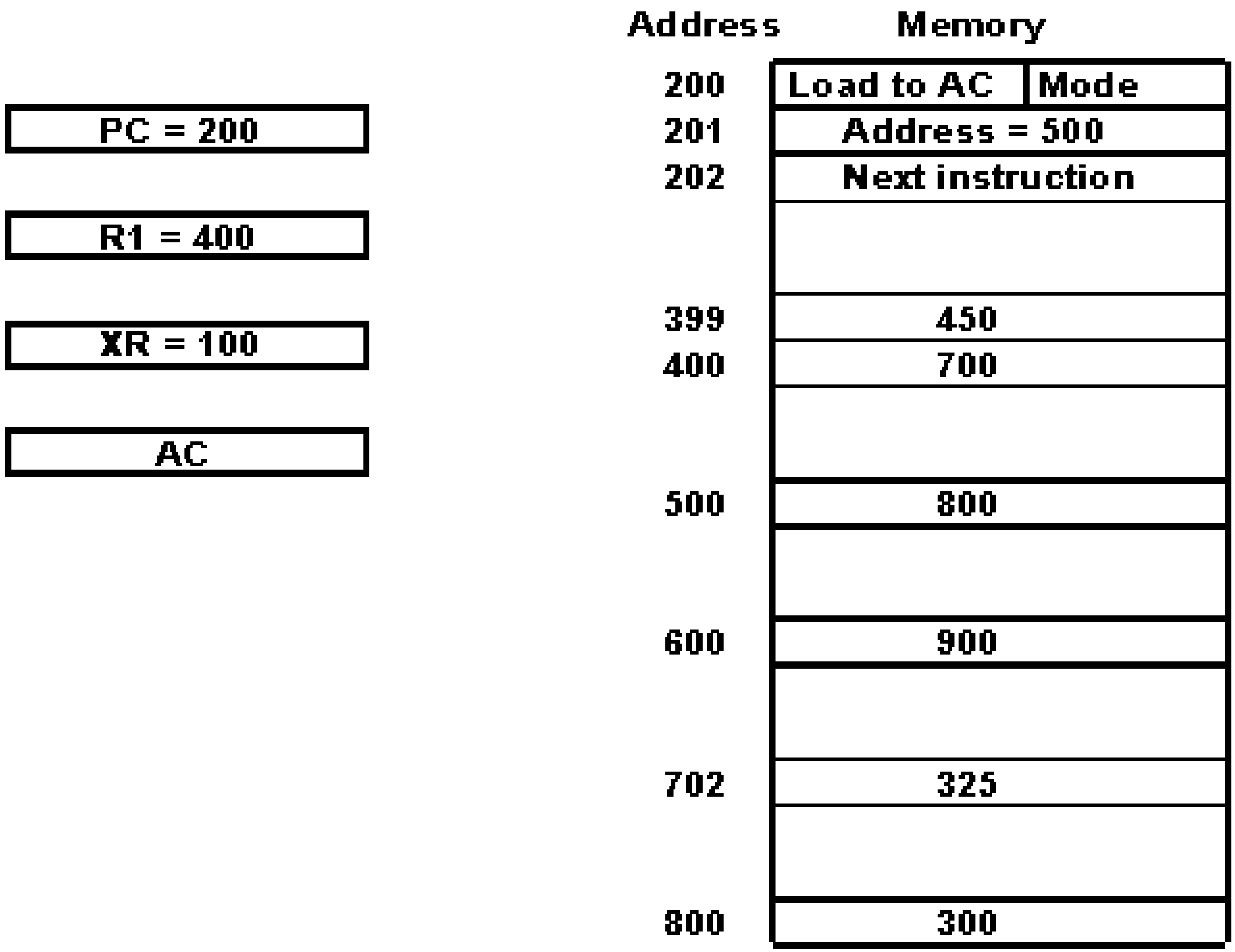
ADDRESS	LABEL	OPCODE	MNEMONICS	OPERAND	COMMENTS
4100		C3	CLR	C	Clear carry
4101		74,04	MOV	A, #DATA1	Move data1 to acc
4103		24,02	ADD	A, #DATA2	Add data2 with acc
4105		90,41,50	MOV	DPTR, #4150h	Move content in 4500 to DPTR.
4108		FO	MOVX	@DPTR, A	Move data to DPTR location
4109		A3	INC	DPTR	Increment DPTR
410A		74,12	MOV	A, #DATA1	Move data1 to acc
410C		34,56	ADDC	A, #DATA2	Add with carry
410E		FO	MOVX	@DPTR, A	Move data to dp location
410F	HERE	80, FE	SJMP	HERE	End of program

Find the Output?

ADDRESS	LABEL	OPCODE	MNEMONICS	OPERAND	COMMENTS
4100		16,00	MVI	D, 00	Clear d register
4102		1E, 00	MVI	E, 00	Clear e register
4104		3A, 53,42	LDA	4253	Load data to acc
4107	HUND	FE, 64	CPI	64H	Compare data with acc
4109		DA, 12,41	JC	TEN	Jump on carry to adder
410C		D6, 64	SUI	64	Subtract data from acc
410E		1C	INR	E	Increment e register
410F		C3, 07,41	JMP	HUND	Jump to address
4112	TEN	FE, 0A	CPI	0AH	Compare data with acc
4114		DA, 1D, 41	JC	UNIT	Jump on carry to adder
4117		D6, 0A	SUI	0AH	Subtract data with acc
4119		14	INR	D	Increment d register
411A		C3, 12,41	JMP	TEN	Jump to address
411D	UNIT	4F	MOV	C, A	Move acc to c register
411E		7A	MOV	A, D	Move data to acc
411F		07	RLC		Rotate left without cy
4120		07	RLC		Rotate left without cy
4121		07	RLC		Rotate left without cy
4122		07	RLC		Rotate left without cy
4123		81	ADD	C	Add data to acc
4124		32,50,42	STA	4250	Store the result
4127		7B	MOV	A, E	Move data to acc
4128		**32, 51,42**	**STA**	**4251**	Store the result

Problems

1. Find the effective address and the content of AC for the given data.



Addressing Mode	Effective Address		Content of AC
Direct Address	**500**	**AC ← (500)**	**800**
Immediate operand	**201**	**AC ← 500**	**500**
Indirect address	**800**	**AC ← ((500))**	**300**
Relative address	**702**	**AC ← (PC + 500)**	**325**
Indexed address	**600**	**AC ← (XR + 500)**	**900**
Register	**-**	**AC ← R1**	**400**
Register Indirect	**400**	**AC ← (R1)**	**700**
Autoincrement	**400**	**AC ← (R1)+**	**700**
Autodecrement	**399**	**AC ← -(R1)**	**450**

- 2. An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is (a) direct; (b) immediate (c) relative (d) register indirect; (e) index with R1 as the index register.

Subroutine Call and Return Mechanisms

Subroutine

- Subroutine is a self-contained sequence of instructions that performs a given computational task.
- It may be called many times at various points in the main program
- When called, branches to 1^{st} line of subroutine and at the end, returned to main program.
- Different names to the instruction that transfers program control to a subroutine
  - Call subroutine
  - Jump to subroutine
  - Branch to subroutine
  - Branch and save address

Control transfer from called to caller

- Subroutine instruction – Opcode + starting address of the subroutine
- Execution:
  - PC content (return address) is stored in a temporary location
  - Control is transferred to the subroutine
- when return
  - Transfers the return address from the temporary location to the PC.
  - Control is transferred back to the called routine

Locations to store the return address

- First memory location of the subroutine
- Fixed location in memory
- Processor registers
- Memory stack – best option
  - Adv: In the case of sequential calls to subroutines. So, the top of the stack always has the return address of the subroutine which to be returned first.

MICRO-OPERATIONS

Call:

SP ← SP – 1 // decrement stack pointer

M[SP] ← PC // push content of PC onto the stack

PC ← effective address /* transfer control to the subroutine */

Return:

PC ← M[SP] // pop stack and transfer to PC

SP ← SP + 1 // increment stack pointer

Recursive subroutines

- Subroutine that calls itself
- If only one register or memory location is used to hold the return address, when subroutine is called recursively, it destroys the previous return address.
- So, stack is the good solution for this problem

ASSIGNMENT

1. Write an assembly language program using IAS instruction set for performing all arithmetic operations (+, -, *, /)
2. Show the register transfer operations using IAS machine registers for division operation.
3. Given the memory contents of the IAS computer shown below. Show the assembly language code for the program, starting at address 08A. Explain what this program does. Given the memory contents of the IAS computer shown below. Show the assembly language code for the program, starting at address 08A. Explain what this program does.

Address	**Contents**
08A	010FA210FB
08B	010FA0F08D
08C	020FA210FB

4. Write an Assembly language programming for the following expressions using IAS computer Instruction set and interpret to the flow of IAS computer [Any one]

   1. A=(B-C)*D
   2. A=B*(C+D)
   3. A=(B-C)/D
   4. A=B/(C+D)
   5. A=-(B+C-D)
   6. A=(B*2)/2

   Make necessary assumptions.

5. On the IAS, describe in English the process that the CPU must undertake to read a value from memory and to write a value to memory in terms of what is put into the MAR, MBR, address bus, data bus, and control bus.

6. Find out the difference between Multicomputer, Multiprocessor, Distributed computer, Multicores

7. A two-word instruction is stored in memory at an address designated by the symbol W. The address field of the instruction (stored at W + 1) is designated by the symbol Y. The operand used during the execution of the instruction is stored at an address symbolized by Z. An index register contains the value X. State how Z is calculated from the other addresses if the addressing mode of the instruction is

- Direct
- Indirect
- Relative
- Indexed

8. A relative mode branch type of instruction is stored in memory at an address equivalent to decimal 750. The branch is made to an address equivalent to decimal 500. What should be the value of the relative address field of the instruction (in decimal)?

9. How many times does the control unit refer to memory when it fetches and executes an indirect addressing mode instruction if the instruction is (a) a computational type requiring an operand from memory; (b) a branch type.

10. What must the address field of an indexed addressing mode instruction be to make it the same as a register indirect mode instruction?

11. An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is (a) direct; (b) immediate (c) relative (d) register indirect; (e) index with R1 as the index register.

12. Assume that in a certain byte-addressed machine all instructions are 32 bits long. Assume the following state of affairs for the machine: Fill in the following table:

Address	Value
PC	100
R0	200
R1	300
100	200
104	300
108	400
200	500
300	600
500	700

Instruction	Addressing mode	Value in R0
Load r0, #200	**Immediate**	
Load r0, 200	**Direct**	
Load r0, (200)	**Indirect**	
Load r0,r1	**Register**	
Load r0, [r1]	**Register Indirect**	
Load r0, -100[r1]	**Based**	
Load r0, 200[PC]	**Relative**	

13. Given the following memory values and a one-address machine with an accumulator, what values do the following instructions load into the accumulator?

- Word 20 contains 40
- Word 30 contains 50
- Word 40 contains 60
- Word 50 contains 70
    - Load immediate 20
    - Load direct 20
    - Load indirect 20
    - Load immediate 30
    - Load direct 30
    - Load indirect 30

14. Let the address stored in the program counter be designated by the symbol X1. The instruction stored in X1 has the address part (operand reference) X2. The operand needed to execute the instruction is stored in the memory word with address X3. An index register contains the value X4. What is the relationship between these various quantities if the addressing mode of the instruction is (a) direct (b) indirect (c) PC relative (d) indexed?

15. An address field in an instruction contains decimal value 14. where is the corresponding operand located for:

- Immediate addressing?
- Direct addressing?
- Indirect addressing?
- Register addressing?
- Register indirect addressing?

16. A PC-relative mode branch instruction is stored in memory at address 620_{10}. The branch is made to location 530_{10}. The address field in the instruction is 10 bits long. What is the binary value in the instruction?

REFERENCES

- William Stallings “Computer Organization and architecture” 8th edition:
  - History of computing :pg 35-56
  - IAS organization : pg 36 -42
  - Instruction fetch & execute : pg 87 – 91
  - Addressing Modes : pg 419 -426

M. M. Mano, Computer System Architecture, Prentice-Hall

Instruction format : pg 255-260

subroutine call & return statement : pg 278 -279

Vincent .P. Heuring, Harry F. Jordan “ Computer System design and Architecture” Pearson, 2nd Edition, 2003

Instruction format calculation