RAMA UNIVERSITY

www.ramauniversity.ac.in

FACULTY OF ENGINEERING & TECHNOLOGY

CSPS-106 Computer Organization

Lecture-16

Mr. Dilip Kumar J Saini
Assistant Professor
Computer Science & Engineering

OUTLINE

- ADDRESSING MODES - EXAMPLES
- DATA TRANSFER INSTRUCTIONS
- DATA MANIPULATION INSTRUCTIONS
- PROGRAM CONTROL INSTRUCTIONS
- CONDITIONAL BRANCH INSTRUCTIONS
- CONDITIONAL BRANCH INSTRUCTIONS

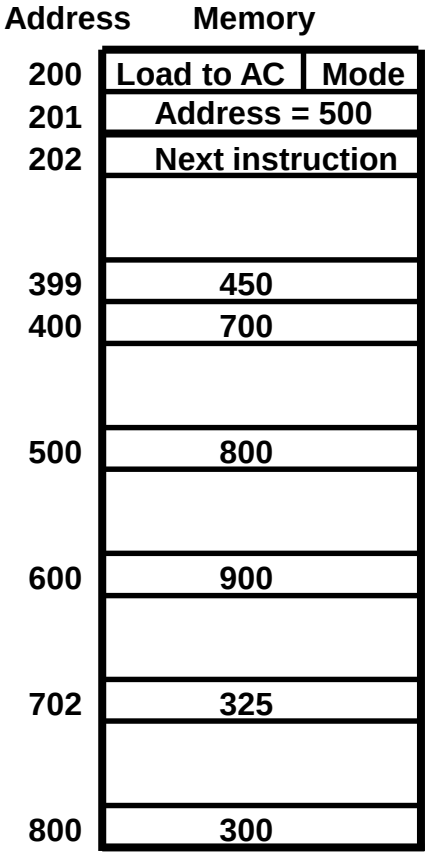
ADDRESSING MODES - EXAMPLES

PC = 200

R1 = 400

XR = 100

AC



Addressing Mode	Effective Address			Content of AC
Direct address	500	/* AC ← (500)	*/	800
Immediate operand	-	/* 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 ← -(R)	*/	450

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
Logical and bit manipulation instructions
Shift instructions

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

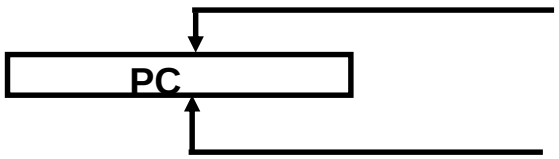
Logical and Bit Manipulation Instructions

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
Disable interrupt	DI

Shift Instructions

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

PROGRAM CONTROL INSTRUCTIONS



+1
In-Line Sequencing
(Next instruction is fetched from the next adjacent location in the memory)

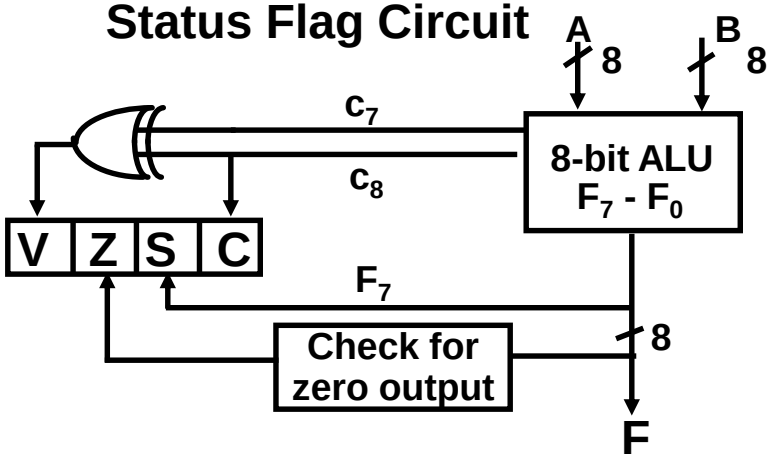
Address from other source; Current Instruction, Stack, etc
Branch, Conditional Branch, Subroutine, etc

Program Control Instructions

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.

Status Flag Circuit



CONDITIONAL BRANCH INSTRUCTIONS

Mnemonic	Branch condition	Tested condition
BZ	Branch if zero	Z = 1
BNZ	Branch if not zero	Z = 0
BC	Branch if carry	C = 1
BNC	Branch if no carry	C = 0
BP	Branch if plus	S = 0
BM	Branch if minus	S = 1
BV	Branch if overflow	V = 1
BNV	Branch if no overflow	V = 0
	Unsigned **compare conditions (A - B)**	
BHI	Branch if higher	A > B
BHE	Branch if higher or equal	A ≥ B
BLO	Branch if lower	A < B
BLOE	Branch if lower or equal	A ≤ B
BE	Branch if equal	A = B
BNE	Branch if not equal	A ≠ B
	Signed **compare conditions (A - B)**	
BGT	Branch if greater than	A > B
BGE	Branch if greater or equal	A ≥ B
BLT	Branch if less than	A < B
BLE	Branch if less or equal	A ≤ B
BE	Branch if equal	A = B
BNE	Branch if not equal	A ≠ B

SUBROUTINE CALL AND RETURN

SUBROUTINE CALL Call subroutine
Jump to subroutine
Branch to subroutine
Branch and save return address

Two Most Important Operations are Implied;

***** Branch to the beginning of the Subroutine
- Same as the Branch or Conditional Branch

* Save the Return Address to get the address of the location in the Calling Program upon exit from the Subroutine
- Locations for storing Return Address:
  - Fixed Location in the subroutine(Memory)
  - Fixed Location in memory
  - In a processor Register
  - In a memory stack
    - most efficient way

CALL
SP ← SP - 1
M[SP] ← PC
PC ← EA

RTN
PC ← M[SP]
SP ← SP + 1

External interrupts

External Interrupts initiated from the outside of CPU and Memory

- I/O Device -> Data transfer request or Data transfer complete

- Timing Device -> Timeout

- Power Failure

Internal interrupts (traps)

Internal Interrupts are caused by the currently running program

- Register, Stack Overflow

- Divide by zero

- OP-code Violation

- Protection Violation

Software Interrupts

Both External and Internal Interrupts are initiated by the computer Hardware.

Software Interrupts are initiated by texecuting an instruction.

- Supervisor Call -> Switching from a user mode to the supervisor mode

-> Allows to execute a certain class of operations

which are not allowed in the user mode

Multiple Choice Question

MUTIPLE CHOICE QUESTIONS:

Sr no	Question	Option A	Option B	OptionC	OptionD
1	ASM-Hwidelyused____ assembler:	S/370	S/380	S/390	S/360
	Assemblerisa_____:	Interpreter	Translator	Exchanger	None of these
2	A _____ processor controls repetitious writing of sequence	MACRO	MICRO	NANO	All of these
3	IBM-360 type language is example which supporting language:	Micro	Macro	Both a &b	None of these
4	_____is attached to using macro instruction definition	Name	Definition	Identifier	All of these
5	END of macro definitionby	NAME	MEND	DATA	MEMORY

REFERENCES

- http://www.engppt.com/search/label/Computer%20Organization%20and%20Architecture
- http://www.engppt.com/search/label/Computer%20Architecture%20ppt