TOPIC:

Program Control, Program Control: Status bits, Conditional Branch Instructions, Subroutines call and return Program Interrupts & Types

Dr. Gaurav Sharma
Associate Professor
DICE-CUIET

Data Transfer and Manipulation

- Instruction set of different computers differ from each other mostly in way the operands are determined from the address and mode fields.

The basic set of operations available in a typical computer are :

- **Data Transfer Instructions**
- **Data Manipulation Instruction** :
  perform arithmetic, logic and shift operation
- **Program Control Instructions**
  decision making capabilities, change the path taken by the program when executed in computer.

Data Transfer Instructions

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

Most common transfer : processor reg -memory, processor reg -I/O,
between processor register themselves

- **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

Some assembly language conventions modify the mnemonic symbol to differentiate between the different addressing modes

- **Data Transfer Instructions with Different Addressing Modes**

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

Data Manipulation Instructions

These instruction performs operation on data and provide the computational capabilities for the computer

- **Three Basic Types:**
  - **Arithmetic instructions**
  - **Logical and bit manipulation instructions**
  - **Shift instructions**

Data Manipulation Instructions

Four basic arithmetic operations : + - * /

- **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

Data Manipulation Instructions

- Logical Instructions perform binary operations on string of bits stored in registers
- Useful for manipulating individual/ group of bits
- Consider each bit separately

- **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

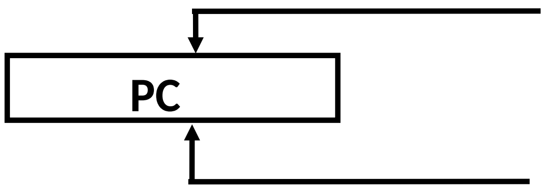
AND	Clear selected bits
OR	Set selected bits
XOR	Complement selected bits

Data Manipulation Instructions

- **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 Instruction



+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**

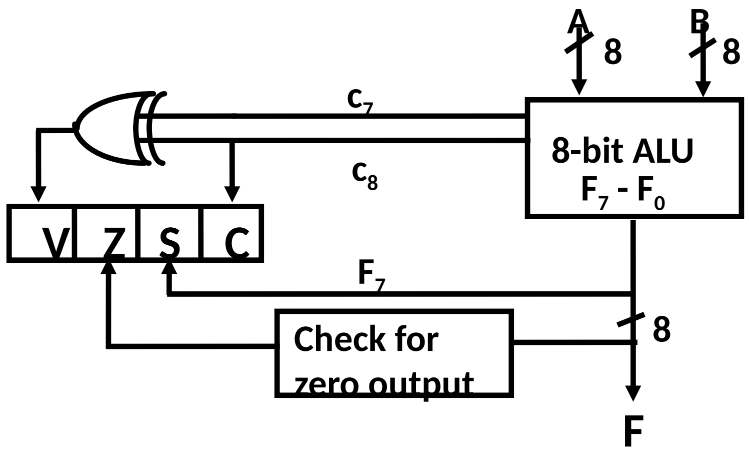
Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RTN
Compare(by -)	CMP
Test(by AND)	TST

*** CMP and TST instructions do not retain their results of operations (- and AND, resp.). They only set or clear certain Flags. like carry /sign/zero bit or overflow condition**

Flag, Processor Status Word

- In Basic Computer, the processor had several (status) flags – 1 bit value that indicated various information about the processor's state – FGI, FGO, I, IEN, R
- In some processors, flags like these are often combined into a register – the processor status register (PSR); sometimes called a processor status word (PSW)
- Common flags in PSW are
  - C (Carry): Set to 1 if the carry out of the ALU is 1
  - S (Sign): The MSB bit of the ALU's output
  - Z (Zero): Set to 1 if the ALU's output is all 0's
  - V (Overflow): Set to 1 if there is an overflow

Status Flag Circuit



Status Bit Condition

Status Bit	Description	Set to 1	Clear to 0
C	Carry	If end carry C8 =1	If carry=0
S	Sign	Highest order bit F7=1	If F7=0
Z	Zero	O/P of ALU contains all 0	otherwise
V	Overflow	EX-OR of last two carries=1	otherwise

Conditional Branch Instruction

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	$A \geq B$
BLO	Branch if lower	$A < B$
BLOE	Branch if lower or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$
	Signed compare conditions (A - B)	
BGT	Branch if greater than	$A > B$
BGE	Branch if greater or equal	$A \geq B$
BLT	Branch if less than	$A < B$
BLE	Branch if less or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$

Subroutine Call and Return

Subroutine : self contained sequence of instructions that performs a given computation task

- **Subroutine Call**
  - ➢ **Call subroutine**
  - ➢ **Jump to subroutine**
  - ➢ **Branch to subroutine**
  - ➢ **Branch and save return address**

- **Instructions are executed by performing two operations :**

  (1). Save PC as Return Address to get the address of the location in the Calling Program upon exit from the Subroutine

  (2). Branch to the beginning of the Subroutine
  - Same as the Branch or Conditional Branch

- **Locations for storing Return Address**
  - **First Location in the subroutine (Memory)**
  - **Fixed Location in memory**
  - **In a processor Register**
  - **In memory *stack***
    - most efficient way

```
CALL
            SP ← SP - 1
            M[SP] ← PC
            PC ← EA

RTN
            PC ← M[SP]
            SP ← SP + 1
```

Interrupt Procedure

Program Interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request

Interrupt Procedure

Interrupt Procedure and Subroutine Call

- The interrupt is usually initiated by an internal or an external signal rather than from the execution of an instruction (except for the software interrupt)

- The address of the interrupt service program is determined by the hardware rather than from the address field of an instruction

- An interrupt procedure usually stores all the information necessary to define the state of CPU rather than storing only the PC.

The state of the CPU is determined from:

- **Content of the PC**
- **Content of all processor registers**
- **Content of status bits**

Many ways of saving the CPU state
depending on the CPU architectures

Interrupt Procedure

- CPU does not respond to interrupt until end of instruction cycle, checks just before next fetch phase

- If interrupt pending, control passed to h/w interrupt cycle:
  - Content of PC and PSW is pushed into stack
  - branch address of interrupt transferred to PC and new PSW loaded into status registers

- Service Program executed starting from branch address and status of CPU as specified by new PSW

- Last instruction from service program is return from interrupt instruction

Program Interrupt

Types of Interrupts

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 supply ㅂ Power Failure

Internal interrupts (traps)

Internal Interrupts are arises from illegal or erroneous use of instn. or data
-ㅂ initiated by program itself rather than external event
- Register, Stack Overflow
- Divide by zero
- OP-code Violation
- Protection Violation

Software Interrupts

Both External and Internal Interrupts are initiated by the computer HW.
Software Interrupts are initiated by the executing an instruction.
- Supervisor Call 1. Switching from a user mode to the supervisor mode
2. Allows to execute a certain class of operations which are not allowed in the user mode