CSE211

Computer Organization and Design

- *Data Transfer and Manipulation*
- *Program Control*
- *RISC and CISC*

Overview

- General Register Organization
- Stack Organization
- Instruction Formats
- Addressing Modes
- **Data Transfer and Manipulation**
- **Program Control**
- **RISC and CISC**

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 ← 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 Maniplulation 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

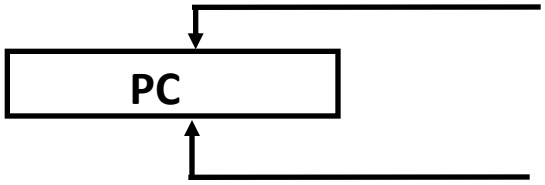
AND →Clear selected bits
OR →Set selected bits
XOR →Complement selected bits

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**

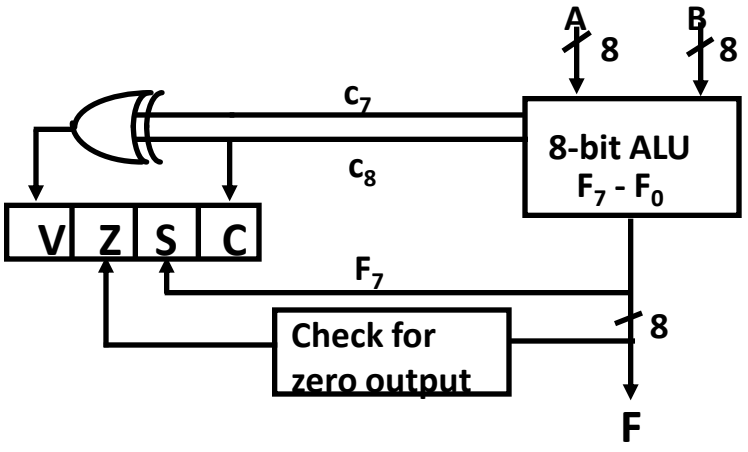
Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RTN
Compare(by –)	CMP
Test(by AND)	TST

*** 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

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 → Power Failure

Internal interrupts (traps)

Internal Interrupts are arises from illegal or erroneous use of instn. or data
-→ 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

CISC

Arguments Advanced at that time

- **Richer instruction sets would simplify compilers**
- **Richer instruction sets would alleviate the software crisis**
  - **move as much functions to the hardware as possible**
- **Richer instruction sets would improve *architecture quality***

CISC

- **These computers with many instructions and addressing modes came to be known as Complex Instruction Set Computers (CISC)**

- **One essential goal for CISC machines was to provide a single machine instruction for each statement that is written in high level language.**
- **Another characteristic was incorporation of variable length instruction format.**

Complex Instruction Set Computers

- **Another characteristic of CISC computers is that they have instructions that act directly on memory addresses**
  - **For example,**

    ADD L1, L2, L3

    that takes the contents of *M*[*L1*] adds it to the contents of *M*[*L2*] and stores the result in location *M*[*L3*]
- **An instruction like this takes three memory access cycles to execute**
- **That makes for a potentially very long instruction execution cycle**
- **The problems with CISC computers are**
  - **The complexity of the design may slow down the processor,**
  - **The complexity of the design may result in costly errors in the processor design and implementation,**
  - **Many of the instructions and addressing modes are used rarely, if ever**

RISC – Reduced Instruction Set Computers

- **In the late ‘70s and early ‘80s there was a reaction to the shortcomings of the CISC style of processors**
- **Reduced Instruction Set Computers (RISC) were proposed as an alternative**
- **The underlying idea behind RISC processors is to simplify the instruction set and reduce instruction execution time**

- **RISC processors often feature:**
  - **Few instructions**
  - **Few addressing modes**
  - **Only load and store instructions access memory**
  - **All other operations are done using on-processor registers**
  - **Fixed length instructions**
  - **Single cycle execution of instructions**
  - **The control unit is hardwired, not microprogrammed**

RISC – Reduced Instruction Set Computers

- **Since all but the load and store instructions use only registers for operands, only a few addressing modes are needed**
- **By having all instructions the same length, reading them in is easy and fast**
- **The fetch and decode stages are simple**
- **The instruction and address formats are designed to be easy to decode**
- **Unlike the variable length CISC instructions, the opcode and register fields of RISC instructions can be decoded simultaneously**
- **The control logic of a RISC processor is designed to be simple and fast**
- **The control logic is simple because of the small number of instructions and the simple addressing modes**
- **The control logic is hardwired, rather than microprogrammed, because hardwired control is faster**