Introduction: Data Transfer and Manipulation

- Most computer instructions can be classified into three categories:
  - 1) Data transfer, 2) Data manipulation, 3) Program control instructions
    - » Data transfer instruction cause transfer of data from one location to another
    - » Data manipulation performs arithmatic, logic and shift operations.
    - » Program control instructions provide decision making capabilities and change the path taken by the program when executed in computer.
- Data Transfer Instruction
  - Typical Data Transfer Instruction :

LD	» **Load** : transfer from memory to a processor register, usually an AC (*memory read*)
ST	» **Store** : transfer from a processor register into memory (*memory write*)
MOV	» **Move** : transfer from one register to another register
XCH	» **Exchange** : swap information between two registers or a register and a memory word
IN/OUT	» **Input/Output** : transfer data among processor registers and input/output device
PUSH/POP	» **Push/Pop** : transfer data between processor registers and a memory stack

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 Address	LD #NBR	AC←NBR
Index Address	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

8 Addressing Mode for the LOAD Instruction

Data Manipulation Instruction

- 1) Arithmetic, 2) Logical and bit manipulation, 3) Shift Instruction
- 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

ADDI Add two binary integer numbers

ADDF Floating point numbers

ADDD ADD TWO DECIMAL NUMBERS IN BCD

- 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 with carry	RORC
Rotate left with carry	ROLC

Program Control

Program Control Instruction :

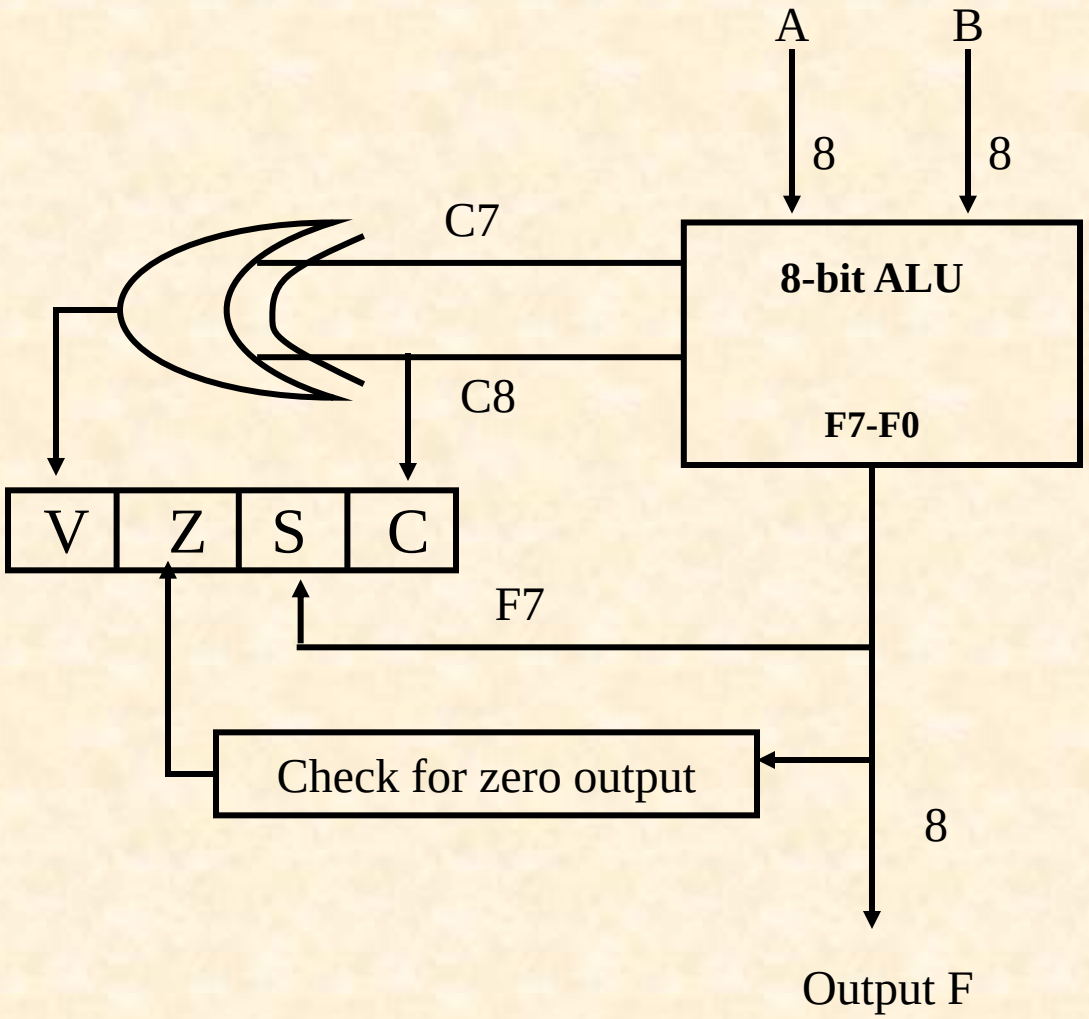
- Branch and Jump instructions are used interchangeably to mean the same thing

NAME	MNEMONIC
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST

◆Status Bit Conditions :

it is sometimes easy to implement the ALU in the CPU with the status register where status bits are used for further analysis.

- ●Condition Code Bit or Flag Bit
  - »The bits are set or cleared as a result of an operation performed in the ALU



- 4-bit status register
  - Bit **C** (*carry*) : set to 1 if the end carry C_8 is 1. It is cleared to 0 if the carry is 0.
  - Bit **S** (*sign*) : set to 1 if the highest order bit F_7 is 1 otherwise 0
  - Bit **Z** (*zero*) : set to 1 if the output of the ALU contains all 0's otherwise 0. In other words Z=1 if the output is zero and z=0 if output is not zero.
  - Bit **V** (*overflow*) : set to 1 if the exclusive-OR of the last two carries (C_8 and C_7) is equal to 1
  - ***Flag Example*** : A - B = A + (2's Comp. Of B) : ***A =11110000, B = 00010100***

11110000
+ 11101100 (2's comp. of B)
1 11011100

C = 1, S = 1, V = 0, Z = 0

Conditional Branch :

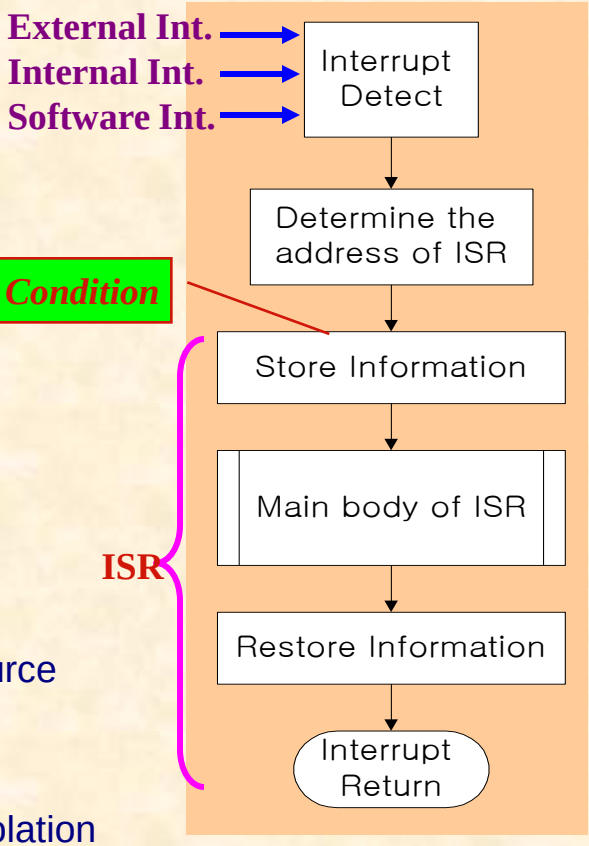
MNEMONIC	Branch Condition	Tested Condition
BZ	BRANCH IF ZERO	Z=1
BNZ	BRANCH IF NOT ZERO	Z=0
BC	BRANCH IF CARRY	C=1
BNC	BRANCH IF NOT CARRY	C=0
	Unsigned compare conditions (A-B)	
BHI	BRANCH IF HIGHER	A>B
BHE	BRANCH IF HIGHER OR EQUAL	A>=B
	signed compare conditions (A-B)	
BGT	BRANCH IF GREATHER THAN	A>B
BGE	BRANCH IF GREATHER OR EQUAL	A>=B

- *Tab. 8-11*
- Subroutine Call and Return
  - CALL :
    $SP \leftarrow SP - 1$: **Decrement stack point**
    $M[SP] \leftarrow PC$: **Push content of PC onto the stack**
    $PC \leftarrow Effective\ \ Address$: **Transfer control to the subroutine**
  - RETURN :
    $PC \leftarrow M[SP]$: **Pop stack and transfer to PC**
    $SP \leftarrow SP + 1$: **Increment stack pointer**
- Program Interrupt
  - Program Interrupt
    - » Transfer program control from a currently running program to another service program as a result of an external or internal generated request
    - » Control returns to the original program after the service program is executed
  - Interrupt Service Program 과 Subroutine Call 의 차이점
    - » 1) An interrupt is initiated by an internal or external signal (*except for software interrupt*)
      - A subroutine call is initiated from the execution of an instruction (CALL)
    - » 2) The address of the interrupt service program is determined by the hardware
      - The address of the subroutine call is determined from the address field of an instruction
    - » 3) An interrupt procedure stores all the information necessary to define the state of the CPU
      - A subroutine call stores only the program counter (*Return address*)

- Program Status Word (PSW)
  - » The collection of all status bit conditions in the CPU
- Two CPU Operating Modes
  - » Supervisor (*System*) Mode : Privileged Instruction 실행
    - When the CPU is executing a program that is part of the operating system
  - » User Mode : User program 실행

PC, CPU Register, Status Condition

CPU operating mode is determined from special bits in the PSW



◆ Types of Interrupts

- 1) External Interrupts
  - » come from I/O device, from a timing device, from a circuit monitoring the power supply, or from any other external source
- 2) Internal Interrupts or TRAP
  - » caused by register overflow, attempt to divide by zero, an invalid operation code, stack overflow, and protection violation
- 3) Software Interrupts
  - » initiated by executing an instruction (**INT** or **RST**)
  - » used by the programmer to initiate an interrupt procedure at any desired point in the program

- Reduced Instruction Set Computer (RISC)
  - Complex Instruction Set Computer (CISC)
    - Major characteristics of a CISC architecture
      - » 1) A large number of instructions - typically from 100 to 250 instruction
      - » 2) Some instructions that perform specialized tasks and are used infrequently
      - » 3) A large variety of addressing modes - typically from 5 to 20 different modes
      - » 4) Variable-length instruction formats
      - » 5) Instructions that manipulate operands in memory (RISC 에서는 in register)
  - Reduced Instruction Set Computer (RISC)
    - Major characteristics of a RISC architecture
      - » 1) Relatively few instructions
      - » 2) Relatively few addressing modes
      - » 3) Memory access limited to load and store instruction
      - » 4) All operations done within the registers of the CPU
      - » 5) Fixed-length, easily decoded instruction format
      - » 6) Single-cycle instruction execution
      - » 7) Hardwired rather than microprogrammed control

- Other characteristics of a RISC architecture
  - » 1) A relatively large number of registers in the processor unit
  - » 2) Use of overlapped register windows to speed-up procedure call and return
  - » 3) Efficient instruction pipeline
  - » 4) Compiler support for efficient translation of high-level language programs into machine language programs

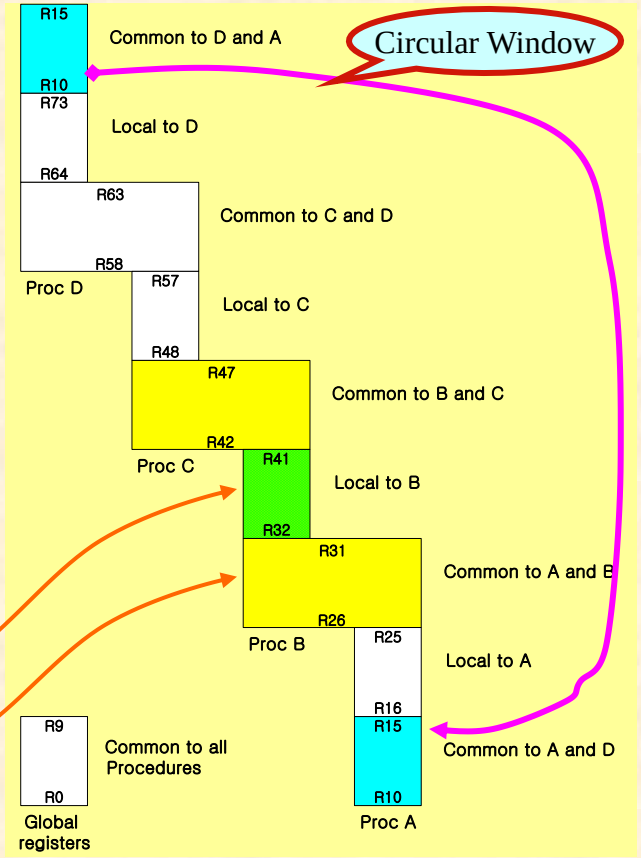
Overlapped Register Windows

- Time consuming operations during procedure call
  - » Saving and restoring registers
  - » Passing of parameters and results
- Overlapped Register Windows
  - » Provide the passing of parameters and avoid the need for saving and restoring register values ***by hardware***

Concept of overlapped register windows : *Fig. 8-9*

- Total 74 registers : R0 - R73
  - » R0 - R9 : Global registers
  - » R10 - R63 : 4 windows
    - Window A
    - Window B
    - Window C
    - Window D

10 Local registers
+
2 sets of 6 registers
(common to adjacent windows)

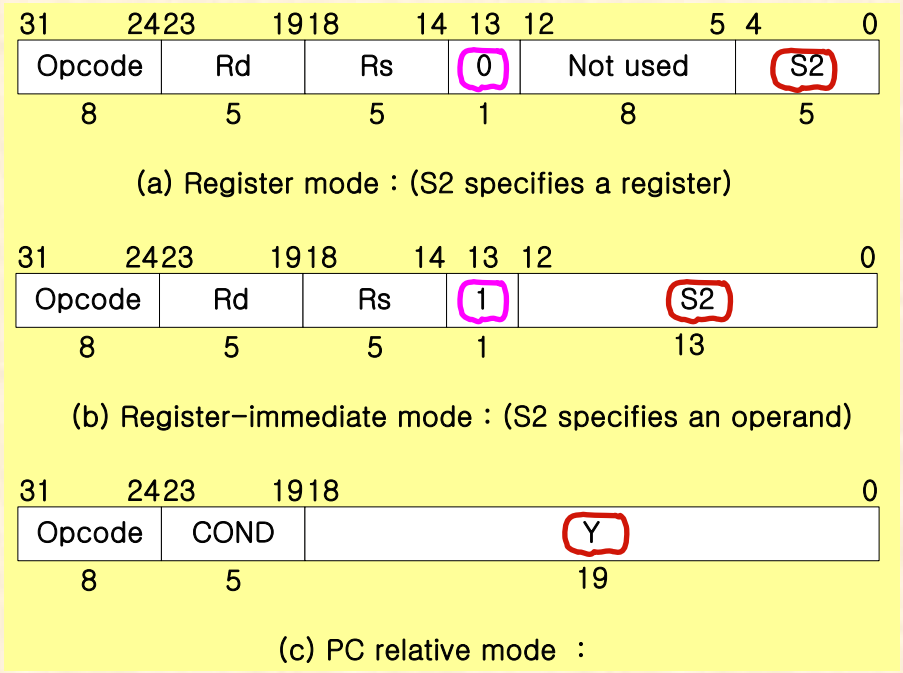


- ***Example)*** Procedure A calls procedure B
  - » R26 - R31
    - Store parameters for procedure B
    - Store results of procedure B
  - » R16 - R25 : Local to procedure A
  - » R32 - R41 : Local to procedure B
- Window Size = L + 2C + G = 10 + (2 X 6) + 10 = 32 registers
- Register File (total register) = (L + C) X W + G = (10 + 6) X 4 + 10 = 74 registers
  - » 여기서, **G** : Global registers = *10*
    L : Local registers = *10*
    C : Common registers = *6*
    W : Number of windows = *4*

◆ Berkeley RISC I

- RISC Architecture 의 기원 : 1980 년대 초
  - » Berkeley RISC project : first project = **Berkeley RISC I**
  - » Stanford MIPS project
- Berkeley RISC I
  - » 32 bit CPU, 32 bit instruction format, 31 instruction
  - » 3 addressing modes : register, immediate, relative to PC

- Instruction Set : ***Tab. 8-12***
- Instruction Format : ***Fig. 8-10***
- Register Mode : ***bit 13 = 0***
  - » S2 = register
  - » ***Example)*** **ADD R22, R21, R23**
    - **ADD Rs, S2, Rd :** Rd = Rs + S2
- Register Immediate Mode : ***bit 13 = 1***
  - » S2 = sign extended 13 bit constant
  - » ***Example)*** **LDL (R22)#150, R5**
    - **LDL (Rs)S2, Rd :** Rd = M[R22] + 150
- PC Relative Mode
  - » Y = 19 bit relative address
  - » ***Example)*** **JMPR COND, Y**
    - Jump to PC = PC + Y
  - » CWP (Current Window Pointer)
    - CALL, RET시 stack pointer 같이 사용됨
- RISC Architecture Originator



Architecture	Originator	Licensees
Alpha	DEC	Mitsubishi, Samsung
MIPS	MIPS Technologies	NEC, Toshiba
PA-RISC	Hewlett Packard	Hitachi, Samsung
PowerPC	Apple, IBM, Motorola	Bul
Sparc	Sun	Fujitsu, Hyundai
i960	Intel	Intel only (Embedded Controller)

Assignment

- What do you mean by Central processing Unit.
- Explain different types of data transfer instructions.