CAO: Lecture 14
RISC, CISC and their comparison II

Topics Covered

- What is RISC?
- OVERLAPPED REGISTER WINDOWS
- CISC versus RISC

What is RISC?

- **RISC?**
  RISC, or *Reduced Instruction Set Computer*. is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures.
- **History**
  The first RISC projects came from IBM, Stanford, and UC-Berkeley in the late 70s and early 80s. The IBM 801, Stanford MIPS, and Berkeley RISC 1 and 2 were all designed with a similar philosophy which has become known as RISC. Certain design features have been characteristic of most RISC processors:
  - *one cycle execution time*: RISC processors have a CPI (clock per instruction) of one cycle. This is due to the optimization of each instruction on the CPU and a technique called PIPELINING
  - *pipelining*: a techique that allows for simultaneous execution of parts, or stages, of instructions to more efficiently process instructions;
  - *large number of registers*: the RISC design philosophy generally incorporates a larger number of registers to prevent in large amounts of interactions with memory

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

– RISC Instruction

- Only use **LOAD** and **STORE** instruction when communicating between memory and CPU
- All other instructions are executed within the registers of the CPU without referring to memory

Program to evaluate *X = (A + B) * (C + D)*

```
LOAD    R1, A
LOAD    R2, B
LOAD    R3, C
LOAD    R4, D
ADD     R1, R1, R2
ADD     R3, R3, R4
MUL     R1, R1, R3
STORE   X, R1
```

$$R1 \leftarrow M[A]$$
$$R2 \leftarrow M[B]$$
$$R3 \leftarrow M[C]$$
$$R4 \leftarrow M[D]$$
$$R1 \leftarrow R1 + R2$$
$$R3 \leftarrow R3 + R4$$
$$R1 \leftarrow R1 * R3$$
$$M[X] \leftarrow R1$$

•Load instruction transfers the operand from memory to CPU Register.

•Add and Multiply operations are executed with data in the registers without accessing the memory.

•Result is then stored in the memory with store information.

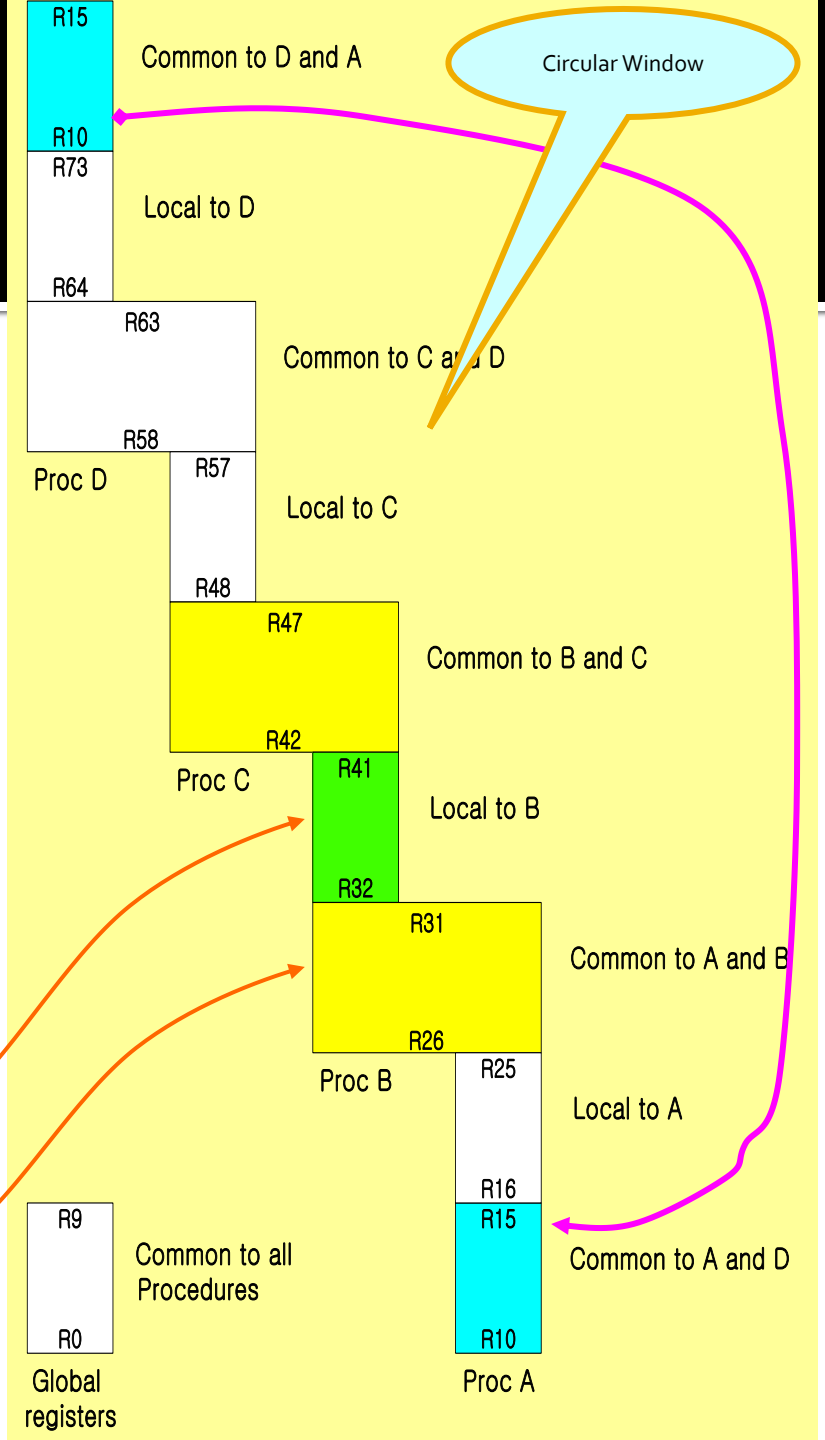
- Other characteristics of a RISC architecture
  - 1) A relatively large number of registers in the processor unit
  - 2) Use of overlapped register windows to speed-up procedure call and return
  - 3) Efficient instruction pipeline
  - 4) Compiler support for efficient translation of high-level language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

OVERLAPPED REGISTER WINDOWS

- There are three classes of registers:
  - Global Registers
    - Available to all functions
  - Window local registers
    - Variables local to the function
  - Window shared registers
    - Permit data to be shared without actually needing to copy it
- Only one register window is active at a time
  - The active register window is indicated by a pointer
- When a function is called, a new register window is activated
  - This is done by incrementing the pointer
- When a function calls a new function, the high numbered registers of the calling function window are shared with the called function as the low numbered registers in its register window
- This way the caller's high and the called function's low registers overlap and can be used to pass parameters and results

- 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)

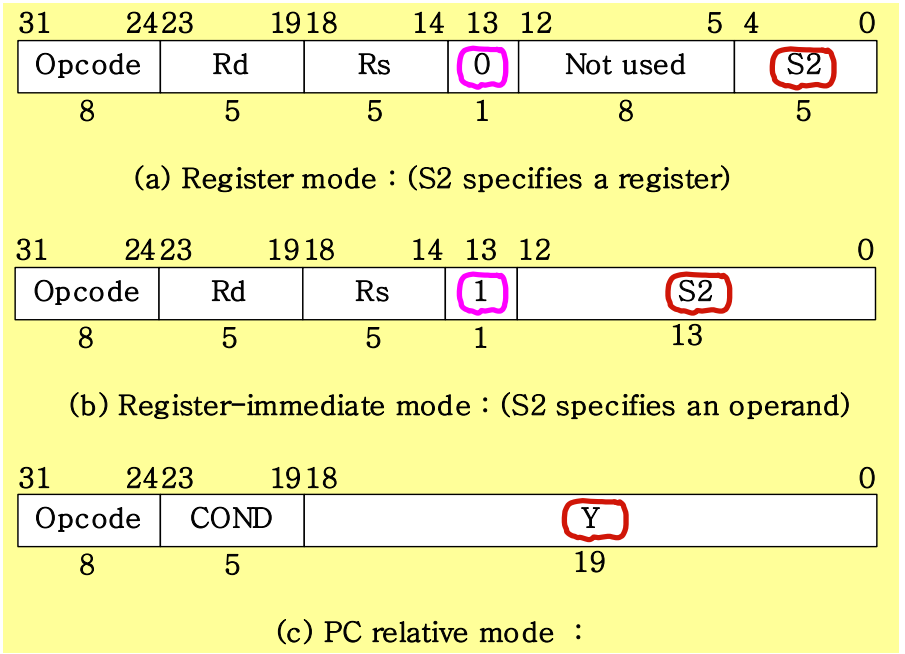


- ***Example)*** Procedure A calls procedure B
  - R26 - R31
    - » Store parameters for procedure B
    - » Store results of procedure B
  - R16 - R25 : Local to procedure A
  - R32 - R41 : Local to procedure B
- Window Size = L + 2C + G = 10 + (2 X 6) + 10 = 32 registers
- Register File (total register) = (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, RET?stack pointer ??⁇



- 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)

CISC versus RISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers