9.2 RISC Machines

- Reduced Instruction Set Computer (**RISC**)
- The underlying philosophy of RISC machines is that a system is better able to manage program execution when the program consists of only a few different instructions that are the same length and require the same number of clock cycles to decode and execute.
- RISC systems access memory only with explicit load and store instructions.
- In CISC systems, many different kinds of instructions access memory, making instruction length variable and fetch-decode-execute time unpredictable.
- The difference between CISC and RISC becomes evident through the basic computer performance equation:

$$\frac{time}{program} = \frac{time}{cycle} \times \frac{cycles}{instruction} \times \frac{instructions}{program}$$

- RISC systems shorten execution time by reducing the clock cycles per instruction.
- CISC systems improve performance by reducing the number of instructions per program.
- The simple instruction set of RISC machines enables control units to be hardwired for maximum speed.
- The more complex-- and variable-- instruction set of CISC machines requires microcode-based control units that interpret instructions as they are fetched from memory. This translation takes time.
- With fixed-length instructions, RISC lends itself to pipelining and speculative execution.
- Consider the program fragments:

mov ax, 10 **mov bx, 5** **mul bx, ax**	**mov ax, 0** **mov bx, 10** **mov cx, 5** **Begin: add ax, bx** **loop Begin**
CISC **32 clock cycles**	**RISC** **13 clock cycles**

- CISC instructions:
  $Total\ clock\ cycles = (2\ movs \times 1\ clock\ cycle) + (1\ mul \times 30\ clock\ cycles)$
  $Total\ clock\ cycles = 32\ clock\ cycles$

- RISC instructions:
  $Total\ clock\ cycles = (3\ movs \times 1\ clock\ cycle) + (5\ adds \times 1\ clock\ cycle) + (5\ loops \times 1\ clock\ cycle)$
  $Total\ clock\ cycles = 13\ clock\ cycles$

- With RISC clock cycle being shorter, RISC gives us much faster execution speeds.

- Because of their load-store ISAs, RISC architectures require a large number of CPU registers.
- These register provide fast access to data during sequential program execution.
- They can also be employed to reduce the overhead typically caused by passing parameters to subprograms.
- Instead of pulling parameters off of a stack, the subprogram is directed to use a subset of registers.
- This is how registers can be overlapped in a RISC system.
- The *current window pointer* (CWP) points to the active register window.

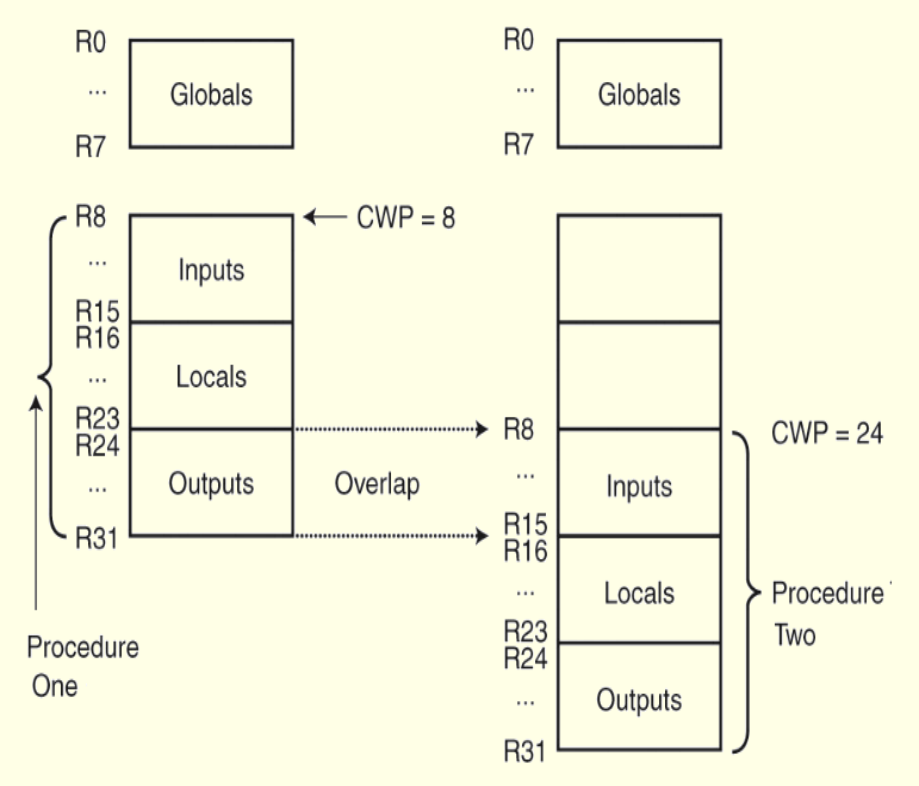
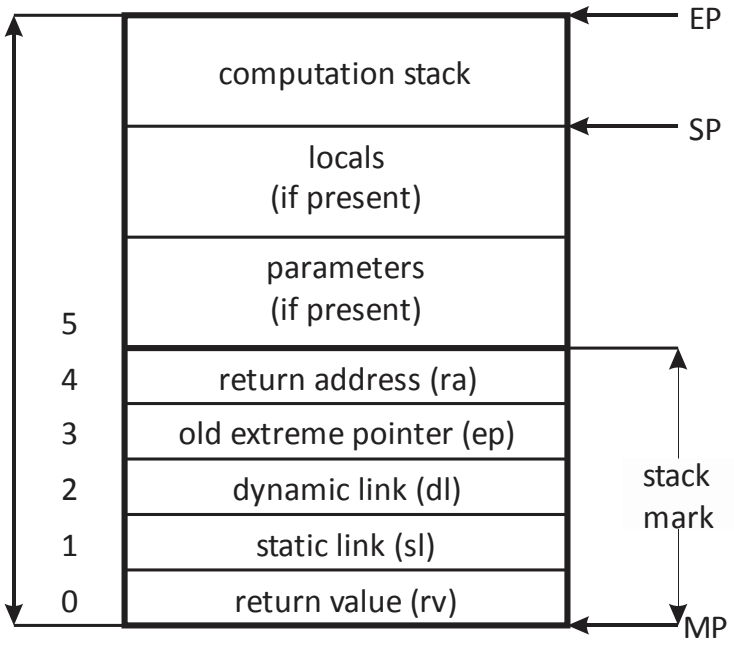
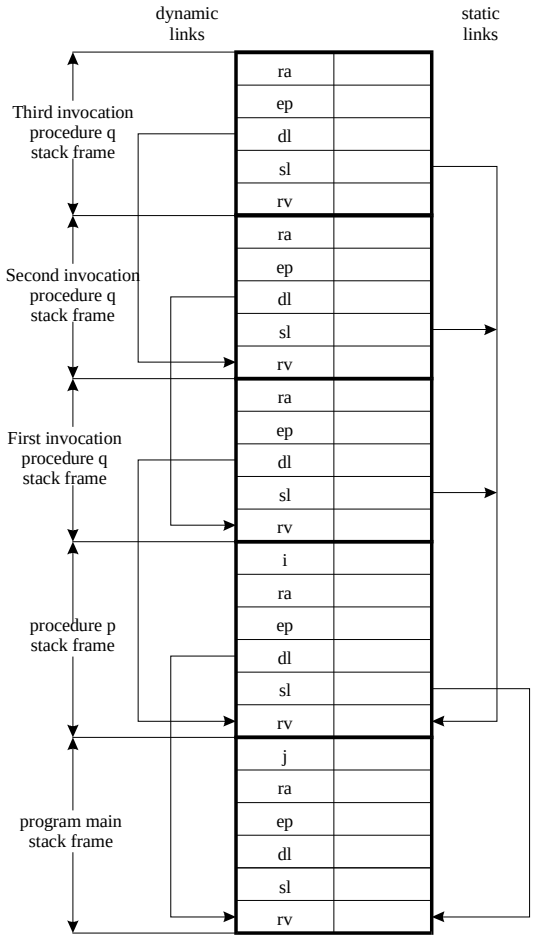


Figure 9.1 Overlapping Register Windows



P-Machine Activation Record

```
program main;
    var j:integer;
    procedure p;
        var i:integer;
        procedure q;
        begin
            if i>0 then
            begin
                i:=i-1;
                j:=j+1;
                q
            end
        end{q};
    begin{p}
        i:=2;
        q;
    end{p};
begin{main}
    j:=0;
    p
end{main}.
```



Pascal Program and Corresponding Activation Records

- It is becoming increasingly difficult to distinguish RISC architectures from CISC architectures.
- Some RISC systems provide more extravagant instruction sets than some CISC systems.
- Some systems combine both approaches.

- The table below summarize the characteristics that traditionally typify the differences between these two architectures.

RISC	CISC
Multiple register sets, often consisting of more than 256 registers	Single register set, typically 6 to 16 registers total
Three register operands allowed per instruction (e.g. `add R1,R2,R3`)	One or two register operands allowed per instruction (e.g. `add R1,R2`)
Parameter passing through efficient on-chip register windows	Parameter passing through inefficient off-chip memory
Single-cycle instructions (except for `load` and `store`)	Multiple-cycle instructions
Hardwired control	Microprogrammed control
Highly pipelined	Less pipelined
Simple instructions that are few in number	Many complex instructions
Fixed-length instructions	Variable-length instructions
Complexity in compiler	Complexity in microcode
Only `load` and `store` instructions can access memory.	Many instructions can access memory
Few addressing modes	Many addressing modes

Table 9.1 The Characteristics of RISC Machines versus CISC Machines