CISC vs RISC
(Y86/X86 vs MIPS)
revisão

Arquitetura de Computadores

Lic. em Engenharia Informática

CISC vs RISC

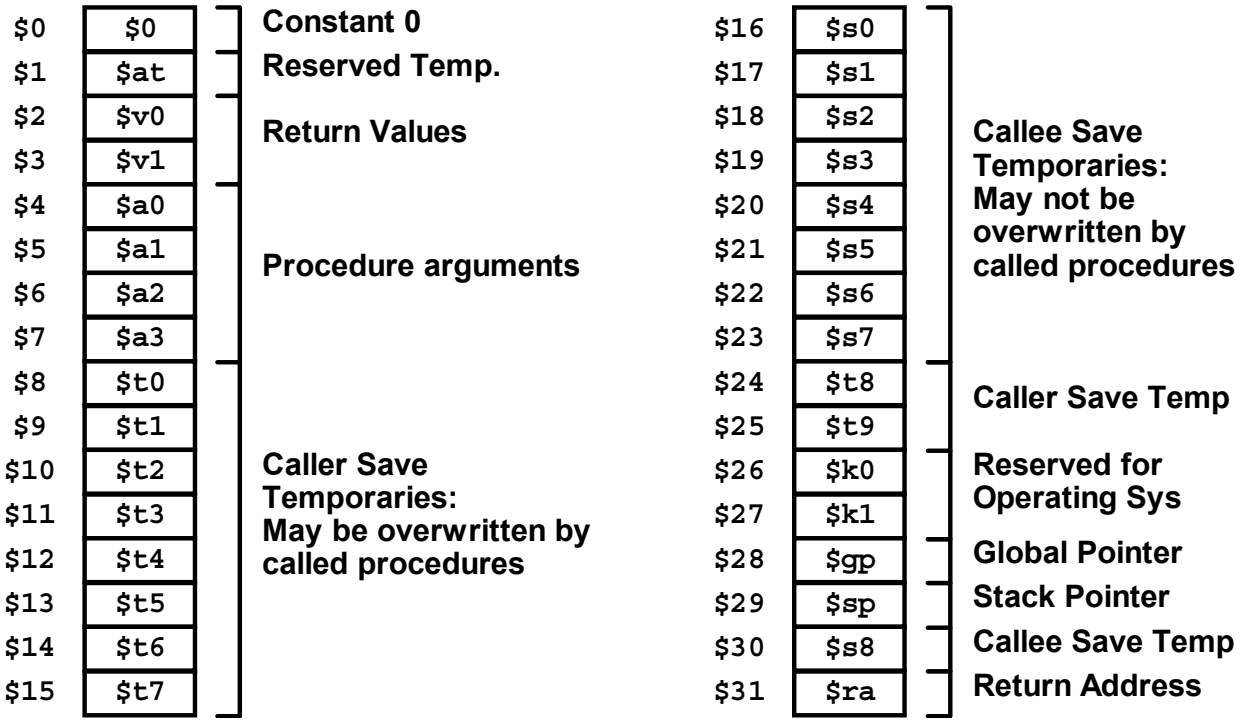
- CISC
  - Conjunto de instruções **complexos**
  - ISA orientado à utilização da **pilha**
    - Usa a pilha para passagem de parâmetros/endereço de retorno
    - Instruções de *push* e *pop*
  - Instruções aritméticas podem aceder à memória
    - addl %eax, 12(%ebx,%ecx,4)
      - Requere uma leitura e uma escrita na memória
      - Aritmética de endereços complexa
  - Variáveis de condição (CC)
    - Ativadas como efeitos colaterais das instruções aritméticas e lógicas

CISC vs RISC

- RISC
  - Conjunto reduzido de **instruções simples**
    - Podem ser necessárias mais instruções
    - Podem ser executadas em hardware simples e mais rápido
  - ISA orientado à utilização da **registos**
    - Elevado número de registos (tipicamente 32)
    - Registos utilizados para argumentos, endereço de retorno, valores temporários
  - Apenas instruções de *load* / *store* acedem à memória
    - Semelhantes às *mrmovl* e *rmmovl* do Y86
  - Não há variáveis de condição (CC)
    - Instruções de teste colocam os valores em registo

CISC vs RISC

- Exemplo de arquitetura RISC: MIPS
  - Registos



CISC vs RISC

- Exemplo de arquitetura RISC: MIPS (cont.)
  - Instruções / Formato de instruções

R-R


```
addu $3,$2,$1        # Register add: $3 = $2+$1
```

R-I


```
addu $3,$2, 3145     # Immediate add: $3 = $2+3145
sll $3,$2,2          # Shift left: $3 = $2 << 2
```

Branch


```
beq $3,$2,dest       # Branch when $3 = $2
```

Load/Store


```
lw $3,16($2)         # Load Word: $3 = M[$2+16]
sw $3,16($2)         # Store Word: M[$2+16] = $3
```

CISC vs RISC

- MIPS vs Y86

	Y86	**MIPS**
Load	mrmovl **D**(**rB**), **rA**	lw **rA**, **D**(**rB**)
Store	rmmovl **rA**, **D**(**rB**)	sw **rA**, **D**(**rB**)
Add, Sub, etc	addl **rA**, **rB**	addu **rD**, **rA**, **rB**
Add c/constante	-	addiu **rD, rA, V**
Imediato-registo	irmovl **rA, V**	addi **rD, $0, V**
Registo-registo	rrmovl **rA, rB**	addu **rD**, **$0**, **rB**
Salto condicional	je dest jnz dest (nota: usa CC)	beq **rA**, **rB**, dest beq **rA**, **$0**, dest

CISC vs RISC

- Debate CISC vs RISC
  - CISC: ISA mais fácil para os compiladores, menos instruções
  - RISC: melhor para as optimizações dos compiladores, execução mais rápida com hardware mais simples
- Estado atual
  - CISC domina nos processadores de secretária
    - Com hardware suficiente um processador CISC pode ser eficiente
    - Compatibilidade do código é importante
      - Internamente são processadores RISC
  - RISC domina nos processadores para sistemas embebidos
    - Mais pequenos, mais baratos, melhor consumo
- Y86 é um ISA posicionado entre RISC e CISC