Arquitetura e Organização de Computadores

Capítulo 13

Computadores com conjunto reduzido de instruções

Principais avanços nos computadores

- Conceito de família:
  - IBM System/360 1964.
  - DEC PDP-8.
  - Separa arquitetura da implementação.
- Unidade de controle microprogramada:
  - Idéia por Wilkes,1951.
  - Produzido por IBM S/360 1964.
- Memória cache:
  - IBM S/360 modelo 85 1969.

- RAM em estado sólido.
  - (Ver notas sobre memória).
- Microprocessadores:
  - Intel 4004 1971.
- Pipelining:
  - Introduz paralelismo no ciclo busca − execução.
- Múltiplos processadores.

O próximo passo − RISC

- Reduced Instruction Set Computer.

- Principais características:
  - Grande número de registradores de uso geral.
  - Ou uso de tecnologia de compilador para otimizar uso do registrador.
  - Conjunto de instruções limitado e simples.
  - Ênfase na otimização do pipeline de instrução.

Comparação de processadores

	Computadores com conjuntos de instruções complexos (CISC)			Computadores com conjuntos de instruções reduzidos (RISC)		Superescalares		
Característica	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000	Power PC	Ultra SPARC	MIPS R10000
Ano de desenvolvimento	1973	1978	1989	1987	1991	1993	1996	1996
Número de instruções	208	303	235	69	94	225		
Tamanho de instrução em bytes	2–6	2–57	1–11	4	4	4	4	4
Modos de endereçamento	4	22	11	1	1	2	1	1
Número de registradores de uso geral	16	16	8	40–520	32	32	40–520	32
Tamanho da memória de controle (Kb)	420	480	246	—	—	—	—	—
Tamanho da cache (KB)	64	64	8	32	128	16–32	32	64

Força motriz para CISC

- Custos de software são superiores aos custos de hardware.
- Linguagens de alto nível cada vez mais complexas.
- Lacuna semântica.
- Leva a:
  - Grandes conjuntos de instruções.
  - Mais modos de endereçamento.
  - Implementações de hardware de instruções de HLL.
    - P.e., CASE (switch) no VAX.

Intenção do CISC

- Fácil escrita de compilador.
- Melhora a eficiência da execução.
  - Operações complexas no microcódigo.
- Admite HLLs mais complexas.

Características de execução

- Operações efetuadas.
- Operandos usados.
- Sequência da execução.
- Estudos têm sido feitos sobre programas escritos em HLLs.
- Estudos dinâmicos são medidos durante a execução do programa.

Operações

- Atribuições:
  - Movimentação de dados.
- Instruções condicionais (IF, LOOP):
  - Controle de sequência.
- Chamada-retorno de procedimento é muito demorado.
- Algumas instruções de HLL ocasionam muitas operações em código de máquina.

Frequência dinâmica relativa ponderada de operações HLL [PATT82a]

	Ocorrência dinâmica		Avaliação das instruções de máquina		Avaliação de referências de memória	
	Pascal	C	Pascal	C	Pascal	C
ATRIBUIÇÃO	45%	38%	13%	13%	14%	15%
LOOP	5%	3%	42%	32%	33%	26%
CHAMADA	15%	12%	31%	33%	44%	45%
IF	29%	43%	11%	21%	7%	13%
GOTO	—	3%	—	—	—	—
OUTROS	6%	1%	3%	1%	2%	1%

Operandos

- Principalmente variáveis escalares locais.
- Otimização deve se concentrar no acesso a variáveis locais.

	Pascal	**C**	**Média**
Constante inteira	16%	23%	20%
Variável escalar	58%	53%	55%
Array/Estrutura	26%	24%	25%

Chamadas de procedimento

- Muito demorada.
- Depende do número de parâmetros passados.
- Depende do nível de aninhamento.
- Maioria dos programas não faz muitas chamadas seguidas por muitos retornos.
- Maioria das variáveis é local.
- (Compare com localidade de referência)

Implicações

- Melhor suporte é dado otimizando recursos mais usados e mais demorados.
- Grande número de registradores:
  - Referência de operando.
- Projeto cuidadoso dos pipelines:
  - Previsão de desvio etc.
- Conjunto de instruções simplificado (reduzido).

Arquivo de registradores grande

- Solução de software:
  - Requer que compilador reserve registradores.
  - Aloca com base nas variáveis mais usadas em determinado momento.
  - Requer análise de programa sofisticada.
- Solução de hardware:
  - Tem mais registradores.
  - Assim, mais variáveis estarão nos registradores.

Registradores para variáveis locais

- Armazenam variáveis escalares locais nos registradores.
- Reduz acesso à memória.
- Cada chamada de procedimento (função) muda a localidade.
- Parâmetros devem ser passados.
- Resultados devem ser retornados.
- Variáveis dos programas que chamam devem ser restauradas.

Janelas de registradores

- Apenas poucos parâmetros.
- Intervalo limitado de profundidade de chamada.
- Usam múltiplos conjuntos pequenos de registradores.
- Chamadas trocam para um conjunto diferente de registradores.
- Retorna a um conjunto de registradores usado anteriormente.

- Três áreas dentro de um conjunto de registradores:
  - Registradores de parâmetros.
  - Registradores locais.
  - Registradores temporários.
  - Registradores temporários de um conjunto sobrepõem registradores de parâmetros do seguinte.
  - Isso permite a passagem de parâmetros sem movimentar dados.

Sobreposição das janelas de registradores

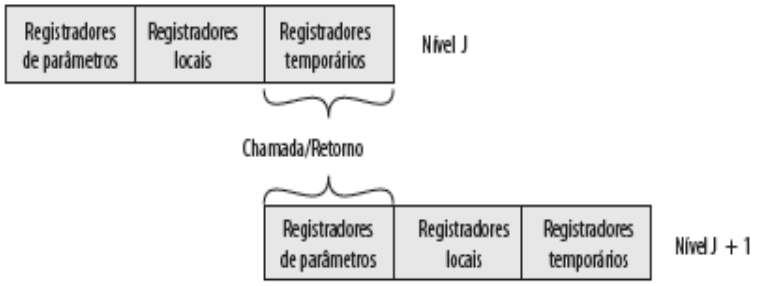
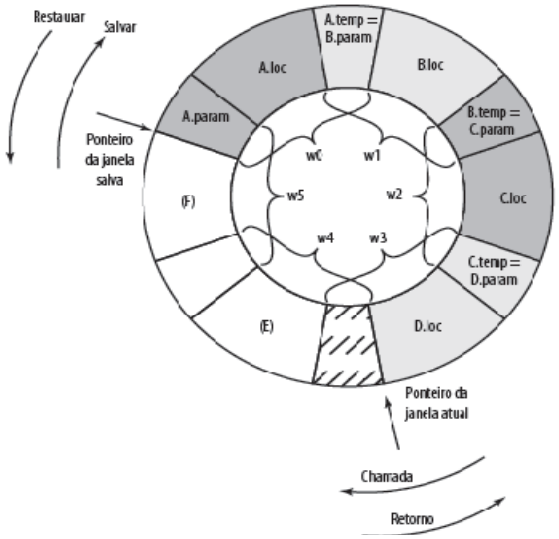


Diagrama de buffer circular



Operação de buffer circular

- Quando uma chamada é feita, um ponteiro de janela atual é movido para mostrar a janela de registrador atualmente ativa.
- Se todas as janelas estiverem em uso, uma interrupção é gerada e a janela mais antiga (aquela mais atrás no aninhamento de chamada) é salva na memória.
- Um ponteiro de janela salva indica para onde as próximas janelas salvas deverão ser restauradas.

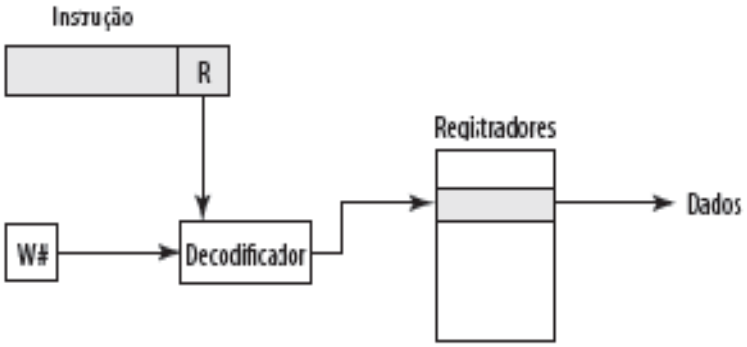
Variáveis globais

- Alocadas pelo compilador para a memória.
  - Ineficaz para variáveis acessadas frequentemente.
- Têm um conjunto de registradores para variáveis globais.

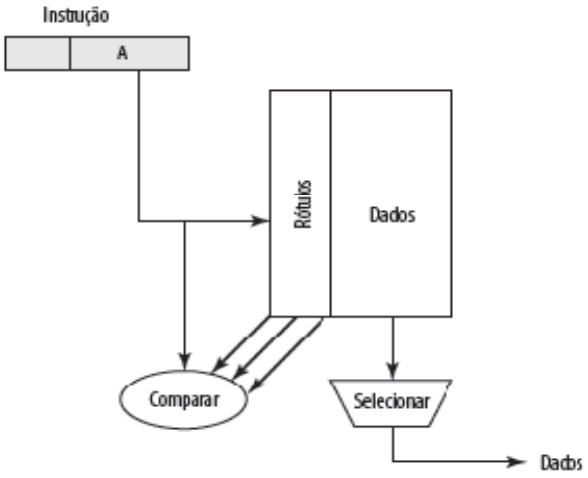
Registradores *versus* Cache

Grandes arquivos de registradores	Cache
Todas variáveis locais escalares	Variáveis locais recentemente usadas
Variáveis individuais	Blocos de memória
Variáveis globais assinaladas pelo compilador	Variáveis globais recentemente usadas
Salvar/restaurar baseados na profundidade de aninhamento do procedimento	Salvar/restaurar baseado em algoritmos de atualização da cache
Endereçamento de registrador	Endereço de memória

Referenciando um escalar – Banco de registradores baseado em janelas



Referenciando um escalar – Cache



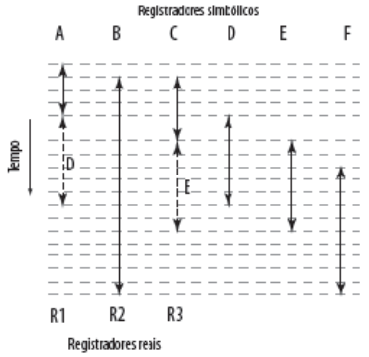
Otimização de registradores baseada em compiladores

- Suponha pequeno número de registradores (16-32).
- Otimizar o uso fica a cargo do compilador.
- Programas HLL não possuem referências a registradores.
  - Normalmente – pense em C – register int.
- Atribua registrador simbólico ou virtual a cada variável candidata.
- Mapeie (ilimitados) registradores simbólicos a registradores reais.
- Registradores simbólicos que não se sobrepõem podem compartilhar registradores reais.
- Se os registradores reais se esgotarem, algumas variáveis utilizam memória.

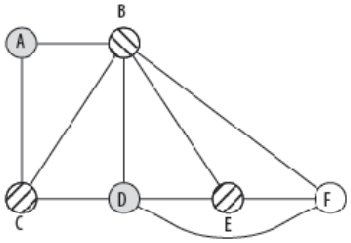
Coloração de grafos

- Dado um grafo de nós e bordas:
- Atribuir uma cor a cada nó.
- Nós adjacentes têm cores diferentes.
- Usar número mínimo de cores.
- Nós são registradores simbólicos.
- Dois registradores que estão vivos no mesmo fragmento de programa são unidos por uma linha.
- Tente colorir o grafo com *n* cores, onde *n* é o número de registradores reais.
- Nós que não podem ser coloridos são colocados na memória.

Abordagem de coloração de grafos



(a) Sequência de tempo do uso ativo de registradores



(b) Grafo de interferência de registradores

Por que CISC?

- Simplificação do compilador?
  - Disputada…
  - Instruções de máquina complexas mais difíceis de explorar..
  - Otimização mais difícil
- Programas menores?
  - Programa ocupa menos memória, mas…
  - Memória agora é barata.
  - Pode não ocupar menos bits, apenas parecer mais curto na forma simbólica.
    - Mais instruções exigem opcodes mais longos.
    - Referências de registrador exigem menos bits.

- Programas mais rápidos?
  - Viés para uso de instruções mais simples.
  - Unidade de controle mais complexa.
  - Maior argumento de controle do microprograma.
  - Assim, instruções simples levam mais tempo para executar.
- Não é tão claro que CISC é a solução apropriada.

Características do RISC

- Uma instrução por ciclo.
- Operações de registrador a registrador.
- Poucos modos de endereçamento simples.
- Poucos formatos de instrução simples.
- Projeto fisicamente conectado (sem microcódigo).
- Formato de instrução fixo.
- Mais tempo/esforço de compilação.

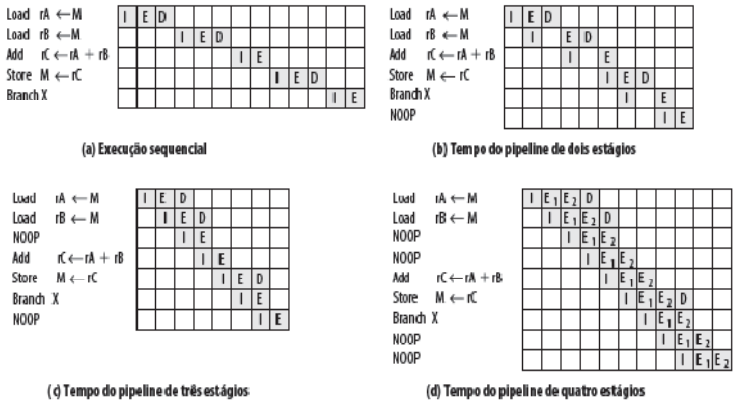
RISC *versus* CISC

- Não bem definido.
- Muitos projetos utilizam ambas as filosofias.
- P.e., PowerPC e Pentium II.

Pipeline no RISC

- Maioria das instruções são de registrador para registrador.
- Duas fases de execução:
  - I: Busca da instrução.
  - E: Execução.
    - Operação da ALU com entrada e saída de registrador.
- Para operações de carregar e armazenar:
  - I: Busca da instrução.
  - E: Execução.
    - Calcular endereço de memória.
  - D: Memória.
    - Operação registrador para memória ou memória para registrador.

Efeitos de pipeline



Optimização de pipeline

- Desvio atrasado:
  - Não tem efeito antes da execução da instrução seguinte.
  - Essa instrução seguinte é o *delay slot.*
- Leitura atrasada:
  - Registrador a ser alvo é bloqueado pelo processador.
  - Continue a execução da instrução até o registrador ser exigido.
  - Ocioso até carga completa.
  - Rearranjar instruções permite que um trabalho útil seja feito enquanto ocorre leitura.
- Laço desenrolado:
  - Replica corpo do laço em um número de vezes.
  - Faz iteração do loop menos vezes.
  - Reduz sobrecarga de laço.
  - Aumenta paralelismo de instrução.
  - Melhor localidade de registradores, cache de dados ou localidade de TLB.

Exemplo de laço desenrolado duas vezes

```
do i=2, n-1
    a[i] = a[i] + a[i-1] * a[i+l]
end do
```

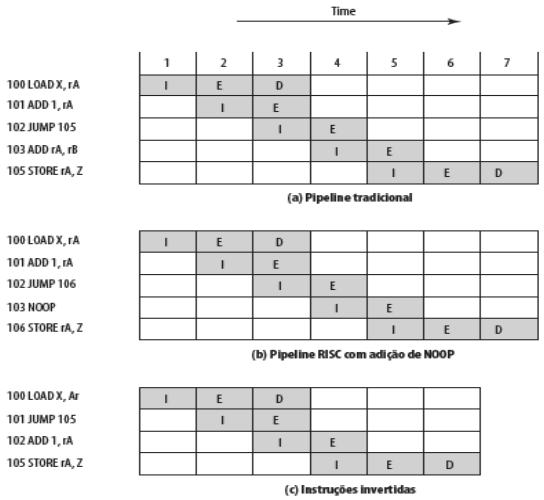
Torna-se:

```
do i=2, n-2, 2
    a[i] = a[i] + a[i-1] * a[i+i]
    a[i+l] = a[i+l] + a[i] * a[i+2]
end do
if (mod(n-2,2) = i) then
    a[n-1] = a[n-1] + a[n-2] * a[n]
end if
```

Desvio normal e atrasado

Endereço	Desvio normal	Desvio atrasado	Desvio atrasado otimizado
100	LOAD X, rA	LOAD X, rA	LOAD X, rA
101	ADD 1, rA	ADD 1, rA	JUMP 105
102	JUMP 105	JUMP 106	ADD 1, rA
103	ADD rA, rB	NOOP	ADD rA, rB
104	SUB rC, rB	ADD rA, rB	SUB rC, rB
105	STORE rA, Z	SUB rC, rB	STORE rA, Z
106		STORE rA, Z	

Uso do desvio atrasado



Controvérsia

- Quantitativa:
  - Compara tamanhos de programa e velocidades de execução.
- Qualitativa.
  - Examina questões de suporte para linguagem de alto nível e uso do estado atual de VLSI.
- Problemas:
  - Não há para de máquinas RISC e CISC comparáveis diretamente.
  - Nenhum conjunto definitivo de programas de teste.
  - Difícil de separar efeitos de hardware dos efeitos do compilador.
  - Maioria das comparações feita sobre máquinas “brinquedo” ao invés de máquinas de produção.
  - Maioria das máquinas comerciais é uma mistura.