RISC - Reduced Instruction Set Computer

MAC 412- Organizãção de Computadores

- Siang W. Song

Baseado no livro de Tanenbaum - Structured Computer Organization

Índice

Críticas à arquitetura CISC

- Os computadores CISC estão ficando demasiadamente complexos, com várias centenas de instruções e dezenas de modos de endereçamento.
- Um grande questionamento é se muitas das intruções complexas são de fato necessárias nos programas “típicos”.
- Levantamentos feitos mostram os comandos mais usados pelos programas. Eles se concentram em alguns poucos comandos, em geral bastante simples.

Levantamento sobre os comandos mais típicos

- Knuth, Wortman, Tanenbaum e Patterson mediram programas escritos em várias linguagens.

Comando	Fortran	C	Pascal
atribuicao :=	51%	38%	45%
if	10	43	29
call	5	12	15
loop	9	3	5
goto	9	3	0
outros	16	1	6

Número de termos	%
1	80
2	15
3	3
4	2
5 ou mais	0

Levantamento sobre uso de variáveis locais e parâmetros

Numero de variaveis locais	%
0	22
1	17
2	20
3	14
4	7
5 ou mais	20

Número parâmetros	%
0	41
1	19
2	15
3	9
4	7
5 ou mais	9

Obervações sobre a velocidade da memória

- Arquitetura CISC se justificava pela velocidade lenta da memória no passado: após precisar de um acesso à memória para buscar uma instrução da memória, a execução das microinstruções resultantes nem sempre precisam de acesso à memória.
- No entanto, as memórias de hoje são bem mais rápidas, e dispensam talvez o uso de uma ROM dentro do processador contendo as microinstruções.

Surgimento da arquitetura RISC

Com as constatações sobre a arquitetura CISC:

- Um microprograma complexo significa maior tempo para decodificar e executar uma instrução complexa, muitas das quais raramente são usadas.
- A vantagem da CISC por causa da memória lenta já não vale diante de memórias modernas mais rápidas.

Surgiu a arquetitura RISC: Reduced Instruction Set Computer.

Característica da arquitetura RISC

- Não há microprograma para interpretar as instruções.
- Existe um conjunto reduzido de instruções simples RISC parecidas com as microinstruções da arquitetura CISC.
- O código gerado pelos compiladores é constituído de instruções simples desse conjunto reduzido. Essas instruções são armazenadas na memória RAM, buscadas e executadas diretamente em hardware na CPU, sem nenhuma interpretação.
- As instruções são executadas na sua maioria em apenas um ciclo da máquina.

Os primeiros computadores RISC

- IBM 801 (1980): é o antecessor do IBM PC/RT (Risc Technology).
- Berkeley RISC I e RISC II (1980 e 1981): projetado por Patterson e Séquin; inspirou o projeto do processador SPARC, da SUN Microsystems.
- Stanford MIPS (1981): projetado por Hennessy; deu origem a MIPS Computer Systems.

Comparação entre CISC e os primeiros RISC

	CISC		RISC		
	IBM 370	VAX 11	IBM 801	RISC I	MIPS
Ano	1973	1978	1980	1981	1983
No. instruções	208	303	120	32	55
Micro código	54K	61K	0	0	0
Instrução (bytes)	2 a 6	2 a 57	4	4	4
Modo endereç.	reg-reg reg-mem mem-mem	reg-reg reg-mem mem-mem	reg-reg	reg-reg	reg-reg

Como é definido o conjunto de instruções RISC

Para definir o conjunto de instruções de um computador RISC, os projetistas usam a seguinte regra de ouro:

- Analisar aplicações para identificar operações-chave.
- Projetar o processador que seja eficiente para essas operações.
- Projetar instruções que realizam as operações-chave.
- Acrescentar mais instruções necessárias, cuidando para não afetar a velocidade da máquina.

Poucos modos de endereçamento

- A maioria das instruções RISC envolvem endereçamento por registrador, sem acesso à memória.
- Portanto as instruções são executadas em um ciclo.
- Pergunta: mas como os valores entram ou saem dos registradores?
  Resposta: há duas instruções LOAD e STORE que acessam a memória.
- Assim, retornamos à origem, como eram os computadores antigos.
- LOAD e STORE em geral levam mais de um ciclo para completar (por envolver acesso a memória).
- Para adiantar as coisas, usa-se o conceito de *pipelining*

Uso de pipelining

- A regra de ouro exige a execução de uma instrução por ciclo.
- No caso de LOAD e STORE, como esses levam mais de um ciclo, a regra de ouro será relaxada.
- Ao invés de exigir a execução de uma instrução por ciclo, será exigido o início da execuçaõ de uma instrução em cada ciclo.
- Suponha que a execução de uma instrução envolve 2 etapas (ou 3 etapas no caso de LOAD e STORE):
  1. busca a instrução da memória
  2. executa a instrução
  3. etapa adicional, no caso de LOAD e STORE
- Cada uma dessas etapas são executadas por circuitos próprios, numa espécie de linha de montagem (pipelining).

Uso de pipelining

Ciclo	1	2	3	4	5	6	7	8	9	10
Busca instrução	1	2	L	4	5	6	S	8	9	10
Execuçaõ instrução		1	2	L	**4**	5	6	S	**8**	9
Ref. memória					L				S	

- No ciclo 1, a instrução 1 é buscada. No ciclo 2, a instrução 2 é buscada e a instrução 1 executada.
- No ciclo 3, a instrução 3 (marcada L para significar LOAD) é buscada e a instrução 2 executada.
- No ciclo 4, a instrução L é iniciada e, como envolve acesso à memória, deve levar mais um ciclo para ser completada.
- No ciclo 5, algo interessante acontece (marcado pelo círculo). A instrução **4** é executada, embora a instrução L não tenha sido completada ainda. Isso é possível desde que a instrução **4** não utilize o registrador que está sendo carregado pela instrução L.

Problema de instruções de desvio

- A execução de instruções por pipelining tem o velho problema do desvio.
- A linha de montagem já preechida precisa ser discartada e busca-se uma nova sequência de instruções para o pipeline.
- Para ter algo a fazer enquanto se busca a nova sequência de instruções por causa de um desvio, o compilador pode colocar após a instrução de desvio uma instrução que logicamente deveria ser executada *antes* do desvio, e executar sempre a instrução seguinte a uma instrução de desvio.
- Note-se assim o importante papel do compilador em máquinas RISC para otimizar os comandos de desvio e, conforme já visto, o uso de comandos LOAD e STORE.

Há muitos registradores na máquina RISC

- O processador RISC não tem microprograma.
- Uma maneira de aproveitar esse espaço é colocar mais registradores.
- Não é incomum ter mais de 500 registradores numa CPU do tipo RISC.
- Parte desses registradores adicioanis pode ser usada para acelerar as chamadas de procedimentos.
- Outro uso dos registradores é para agilizar a mudança de contexto em interrupções.

Usar em chamada de procedimentos e interrupções

- Quando um procedimento é chamado, uma pilha em geral é usada para guardar o ponto de retorno, os parâmetros e as variávies locais.
- Quando dispomos de muitos registradores, essas informações podem ser armazenadas em registradores que atuam como uma pilha dentro do processador.
- Outro uso dos registradores é guardar o contexto de um processo quando ele é suspenso em uma interrupção. Um conjunto de registradores seriam usados, cada um guarda um contexto diferente. Basta um ponteiro para indicar o contexto atual.