

Arquitetura e Organização de Computadores

Aula-10: Conjunto de Instruções: Características e Funções

Eliseu César Miguel

Livro: Arquitetura e Organização de Computadores
William Stallings 8^a Edição
Universidade Federal de Alfenas

February 8, 2021



Organização da Aula

- 1 Introdução
- 2 Características das instruções de máquina
- 3 Tipos de operandos
- 4 Tipos de operações

Organização da Aula

- 1 Introdução
- 2 Características das instruções de máquina
- 3 Tipos de operandos
- 4 Tipos de operações

Organização da Aula

- 1 Introdução
- 2 Características das instruções de máquina
- 3 Tipos de operandos
- 4 Tipos de operações

Organização da Aula

- 1 Introdução
- 2 Características das instruções de máquina
- 3 Tipos de operandos
- 4 Tipos de operações

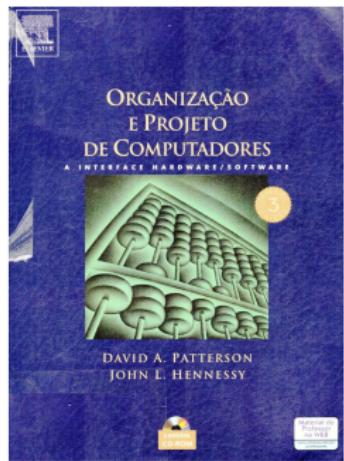
Bibliografia básica



Livro Texto



Complementar



Complementar

Além da bibliografia básica, visite o programa de ensino para ver outras bibliografias. Também, durante o curso, várias bibliografias em sítios da Internet serão apresentadas.

Introdução

Principais pontos sobre Conjunto de Instruções

- Elementos essenciais da instrução: Código de operação (*opcode*); referências a operandos; endereço da próxima instrução
 - ▶ Os *opcodes* definem operações nas categorias de instruções: lógica/aritmética; movimentação de dados; controle
 - ▶ As referências aos operandos podem acessar dados na memória ou registradores, de acordo com o modo de endereçamento
 - ▶ A próxima instrução a ser executada pode ser endereçada de forma implícita ou explícita
- Um recurso implementado nas CPUs é o uso de pilhas para chamadas e retornos de procedimentos (**não se trata de arquitetura de pilha!**)
- Endereçamentos por *byte* podem ser dos tipos: *big-endian*, *little-endian* ou *bi-endian*

Os aspectos discutidos são os mais visíveis ao programador.

Segundo Stallings, *Um limite onde o projetista de computador e o programador de computador podem ver a mesma máquina é o conjunto de instruções de máquina*

Características das instruções de máquina

Um operando pode estar localizado em uma das localidades a seguir:

- Memória principal ou virtual (o endereço real exige muitos *bits* pelo tamanho da memória)
- Registrador do processador (podem ser endereçados por poucos *bits*)
- Imediato (o valor do operando é escrito na própria instrução)
- Dispositivo de E/S (instrução precisa especificar o módulo e o dispositivo)
Para E/S mapeada em memória, o endereço é similar ao endereço de memória

Com o exposto, podemos imaginar que várias técnicas para se determinar um endereço *físico* a partir de um *virtual* devem ser definidas. A isso, damos o nome de *modos de endereçamento*.

Por exemplo, um endereço de uma variável do tipo ponteiro exige duas referências à memória para se localizar o valor do operando.

Geralmente, os compiladores endereçam as instruções do programa de forma relativa ao registrador PC. Com isso, não importa o endereço físico em que o Sistema Operacional irá carregar o programa.

Representação das instruções: As instruções são armazenadas no computador em formato binário. Mnemônicos são atribuídos às instruções para facilitar a visualização e implementação de programas em linguagens de baixo nível.

Exemplo: programação usando o conjunto de instruções do IAS.

Características das instruções de máquina

Representação das instruções: Os vários *bits* da instrução são interpretados pela CPU para compor os campos da instrução. Um mesmo conjunto de instrução pode conter instruções com formatos diferentes. Com isso, a decodificação necessita identificar o formato da instrução.

- IAS: instruções de tamanho fixo e com apenas um formato
- MIPS: instruções de tamanho fixo e com três formatos (RISC)
 - ▶ instruções de 32 bytes;
 - ▶ Registrador/Registrador;
 - ▶ poucos modos de endereçamento
- x86: instruções de tamanho variado e com vários formatos (CISC)
 - ▶ instruções de 1 byte a 18 bytes;
 - ▶ Registrador/Memória;
 - ▶ muitos modos de endereçamento;

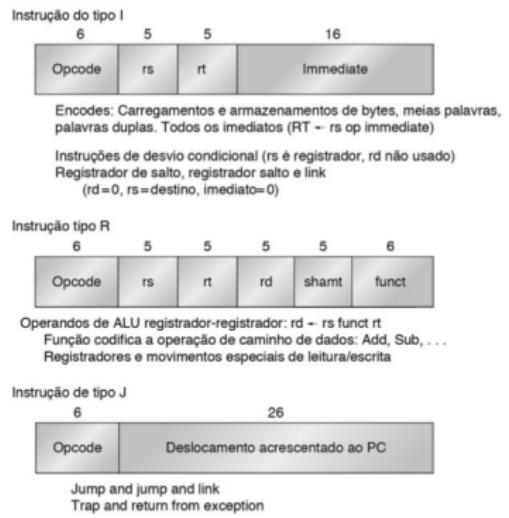


FIGURA A.22 Leiaute de instrução para MIPS.

Todas as instruções são codificadas em um de três tipos, com campos comuns no mesmo local em cada formato.

Características das instruções de máquina

Representação das instruções: Os vários *bits* da instrução são interpretados pela CPU para compor os campos da instrução. Um mesmo conjunto de instrução pode conter instruções com formatos diferentes. Com isso, a decodificação necessita identificar o formato da instrução.

- IAS: instruções de tamanho fixo e com apenas um formato
- MIPS: instruções de tamanho fixo e com três formatos (RISC)
 - ▶ instruções de 32 bytes;
 - ▶ Registrador/Registrador;
 - ▶ poucos modos de endereçamento
- x86: instruções de tamanho variado e com vários formatos (CISC)
 - ▶ instruções de 1 byte a 18 bytes;
 - ▶ Registrador/Memória;
 - ▶ muitos modos de endereçamento;

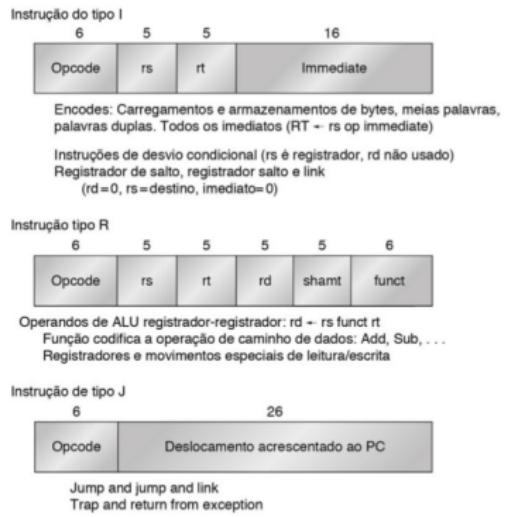


FIGURA A.22 Leiaute de instrução para MIPS.

Todas as instruções são codificadas em um de três tipos, com campos comuns no mesmo local em cada formato.

Características das instruções de máquina

Representação das instruções: Os vários *bits* da instrução são interpretados pela CPU para compor os campos da instrução. Um mesmo conjunto de instrução pode conter instruções com formatos diferentes. Com isso, a decodificação necessita identificar o formato da instrução.

- IAS: instruções de tamanho fixo e com apenas um formato
- MIPS: instruções de tamanho fixo e com três formatos (RISC)
 - ▶ instruções de 32 bytes;
 - ▶ Registrador/Registrador;
 - ▶ poucos modos de endereçamento
- x86: instruções de tamanho variado e com vários formatos (CISC)
 - ▶ instruções de 1 byte a 18 bytes;
 - ▶ Registrador/Memória;
 - ▶ muitos modos de endereçamento;

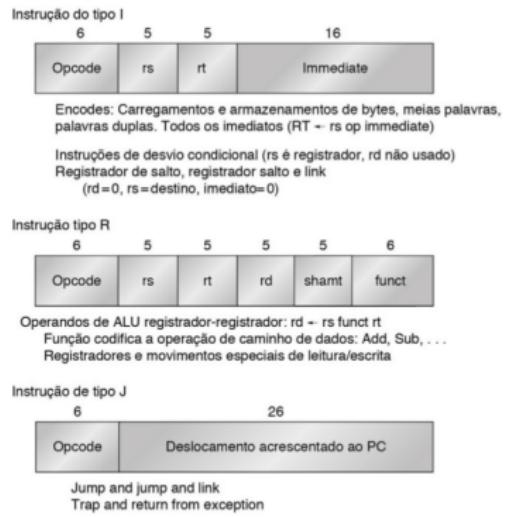


FIGURA A.22 Leiaute de instrução para MIPS.

Todas as instruções são codificadas em um de três tipos, com campos comuns no mesmo local em cada formato.

Características das instruções de máquina

Representação das instruções: Os vários *bits* da instrução são interpretados pela CPU para compor os campos da instrução. Um mesmo conjunto de instrução pode conter instruções com formatos diferentes. Com isso, a decodificação necessita identificar o formato da instrução.

- IAS: instruções de tamanho fixo e com apenas um formato
- MIPS: instruções de tamanho fixo e com três formatos (RISC)
 - ▶ instruções de 32 bytes;
 - ▶ Registrador/Registrador;
 - ▶ poucos modos de endereçamento
- x86: instruções de tamanho variado e com vários formatos (CISC)
 - ▶ instruções de 1 byte a 18 bytes;
 - ▶ Registrador/Memória;
 - ▶ muitos modos de endereçamento;

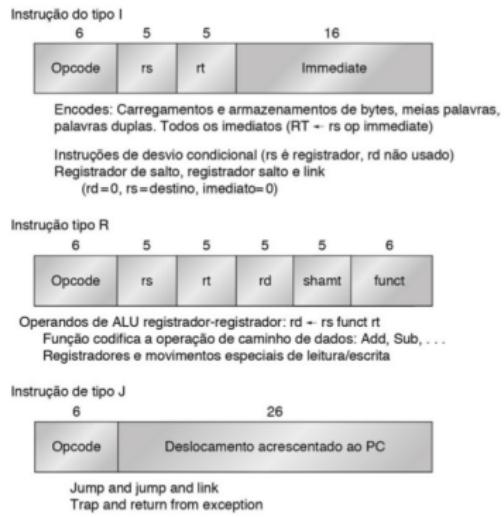


FIGURA A.22 Leiaute de instrução para MIPS.

Todas as instruções são codificadas em um de três tipos, com campos comuns no mesmo local em cada formato.

Características das instruções de máquina

Representação das instruções: Os vários *bits* da instrução são interpretados pela CPU para compor os campos da instrução. Um mesmo conjunto de instrução pode conter instruções com formatos diferentes. Com isso, a decodificação necessita identificar o formato da instrução.

- IAS: instruções de tamanho fixo e com apenas um formato
- MIPS: instruções de tamanho fixo e com três formatos (RISC)
 - ▶ instruções de 32 bytes;
 - ▶ Registrador/Registrador;
 - ▶ poucos modos de endereçamento
- x86: instruções de tamanho variado e com vários formatos (CISC)
 - ▶ instruções de 1 byte a 18 bytes;
 - ▶ Registrador/Memória;
 - ▶ muitos modos de endereçamento;

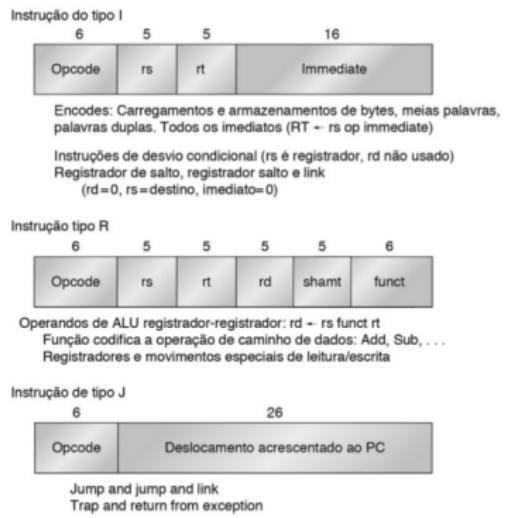


FIGURA A.22 Leiaute de instrução para MIPS.

Todas as instruções são codificadas em um de três tipos, com campos comuns no mesmo local em cada formato.

Características das instruções de máquina

Representação das instruções: Os vários *bits* da instrução são interpretados pela CPU para compor os campos da instrução. Um mesmo conjunto de instrução pode conter instruções com formatos diferentes. Com isso, a decodificação necessita identificar o formato da instrução.

- IAS: instruções de tamanho fixo e com apenas um formato
- MIPS: instruções de tamanho fixo e com três formatos (RISC)
 - ▶ instruções de 32 bytes;
 - ▶ Registrador/Registrador;
 - ▶ poucos modos de endereçamento
- x86: instruções de tamanho variado e com vários formatos (CISC)
 - ▶ instruções de 1 byte a 18 bytes;
 - ▶ Registrador/Memória;
 - ▶ muitos modos de endereçamento;

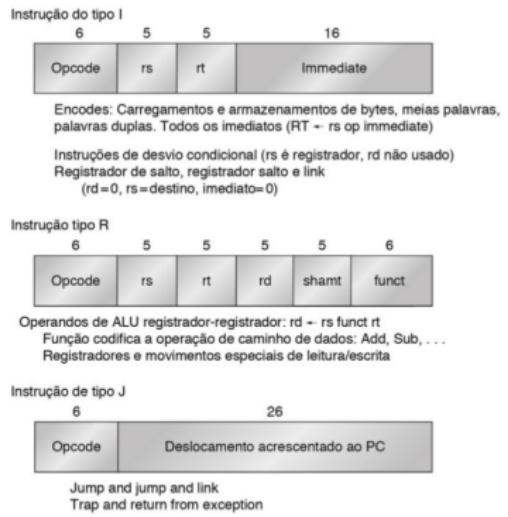


FIGURA A.22 Leiaute de instrução para MIPS.

Todas as instruções são codificadas em um de três tipos, com campos comuns no mesmo local em cada formato.

Características das instruções de máquina

Representação das instruções: Os vários *bits* da instrução são interpretados pela CPU para compor os campos da instrução. Um mesmo conjunto de instrução pode conter instruções com formatos diferentes. Com isso, a decodificação necessita identificar o formato da instrução.

- IAS: instruções de tamanho fixo e com apenas um formato
- MIPS: instruções de tamanho fixo e com três formatos (RISC)
 - ▶ instruções de 32 bytes;
 - ▶ Registrador/Registrador;
 - ▶ poucos modos de endereçamento
- x86: instruções de tamanho variado e com vários formatos (CISC)
 - ▶ instruções de 1 byte a 18 bytes;
 - ▶ Registrador/Memória;
 - ▶ muitos modos de endereçamento;

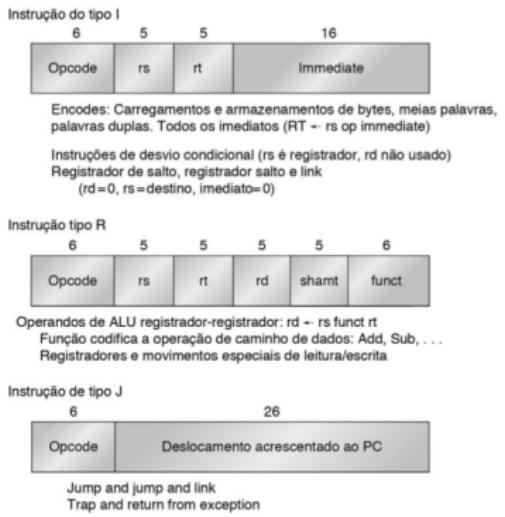


FIGURA A.22 Leiaute de instrução para MIPS.

Todas as instruções são codificadas em um de três tipos, com campos comuns no mesmo local em cada formato.

Características das instruções de máquina

Tipos de instrução: Você pode encontrar diferentes definições na literatura

- processamento de dados: instruções aritméticas e lógicas
- armazenamento e carregamento de dados: movimentação de dados para dentro ou fora do registrador e/ou locais de memória
- movimentação de dados: instruções de E/S
- controle: instruções de teste e desvio

O estudo da presença de instruções por classes nos programas sugere melhorias realizadas no processador para aumentar o desempenho na execução de programas

Isso está, direta ou indiretamente, relacionado a:

- técnicas de predição de desvios
- aumento de unidades funcionais
- número de registradores de propósito geral
- repertório de instruções

Características das instruções de máquina

Número de operandos: Em geral, as arquiteturas podem ter de zero a três referências a operandos em suas instruções. Este valor é definido pelas instruções de lógica/aritmética

Figura 10.3 Programas para executar $Y = \frac{A - B}{C + (D \times E)}$

Instrução	Comentário
SUB Y,A,B	$Y \leftarrow A - B$
MPY T,D,E	$T \leftarrow D \times E$
ADD T,T,C	$T \leftarrow T + C$
DIV Y,Y,T	$Y \leftarrow Y \div T$

(a) Instruções com três endereços

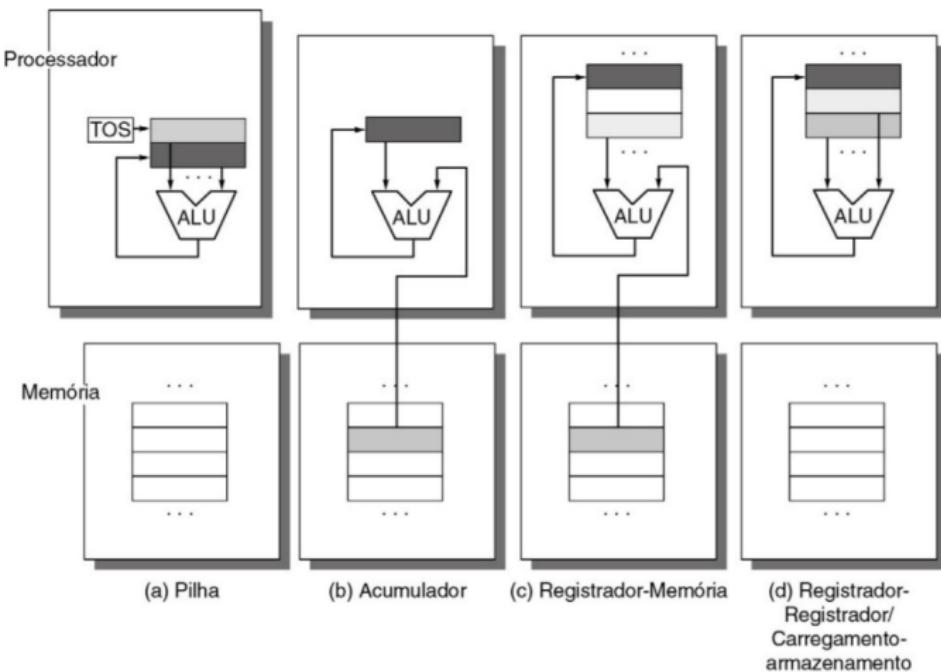
Instrução	Comentário
MOVE Y,A	$Y \leftarrow A$
SUB Y,B	$Y \leftarrow Y - B$
MOVE Y,D	$T \leftarrow D$
MPY T,E	$T \leftarrow T \times E$
ADD T,C	$T \leftarrow T + C$
DIV Y,T	$Y \leftarrow Y \div T$

(b) Instruções de dois endereços

Instrução	Comentário
LOAD D	$AC \leftarrow D$
MPY E	$AC \leftarrow AC \times E$
ADD C	$AC \leftarrow AC + C$
STOR Y	$Y \leftarrow AC$
LOAD A	$AC \leftarrow A$
SUB B	$AC \leftarrow AC - B$
DIV Y	$AC \leftarrow AC \div T$
STOR Y	$Y \leftarrow AC$

(c) Instruções de um endereço

Evolução: Classificação de arquiteturas (operandos)



Taxionomia das memórias: Pilha

Pilha	Acumulador	Registrador (registrador-memória)	Registrador (carregamento-armazenamento)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

Pilha: Não há endereçamento explícito

- ① [Push X] Operandos (memória) são carregados em registradores da ULA
- ② [Add] Operação (memória) é buscada e realizada na ULA
- ③ [Pop C] Resultado é carregado em registrador da ULA
- ④ *Volta a 1 e ciclo recomeça até fim de programa*



Taxionomia das memórias: Acumulador

Pilha	Acumulador	Registrador (registrador-memória)	Registrador (carregamento-armazenamento)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

Acumulador: AC Implícito e operndos de memória explícitos

- ① [Load A] Operando 1 da memória é carregado em AC ($AC \leftarrow M(A)$)
- ② [Add B] Operação é realizada ($AC \leftarrow AC + M(B)$)
- ③ [Stor C] Resultado é carregado na memória ($M(C) \leftarrow AC$)
- ④ Continua ciclo de execução a partir do PC



Arquitetura de Computadores:
Uma abordagem Quantitativa 5ª Edição.
Apêndice A, página A-3



Taxionomia das memórias: Registrador/Memória

Pilha	Acumulador	Registrador (registrador-memória)	Registrador (carregamento-armazenamento)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

Registrador/Memória: Endereços (MEM e REGs) Explícitos

- ① [Load R1, A] Operando da memória é carregado em $R1$ ($R1 \leftarrow M(A)$)
- ② [Add R3, R1, B] Operação é realizada ($R3 \leftarrow R1 + M(B)$)
- ③ [Stor R3, C] Resultado é carregado na memória ($M(C) \leftarrow R3$)
- ④ *Continua ciclo de execução a partir do PC*



Taxionomia das memórias: Registrador/Registrador

Pilha	Acumulador	Registrador (registrador-memória)	Registrador (carregamento-armazenamento)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R3,R1,B	Load R2,B
Add	Store C	Store R3,C	Add R3,R1,R2
Pop C			Store R3,C

Registrador/Registrador: Endereços (MEM e REGs) Explícitos

- ① [Load R1, A] e [Load R2, B] Operandos da memória carregados em registrador
- ② [Add R3, R1, R2] Operação é realizada ($R3 \leftarrow R1 + R2$)
- ③ [Stor R3, C] Resultado é carregado na memória ($M(C) \leftarrow R3$)
- ④ Continua ciclo de execução a partir do PC



Taxionomia das memórias: Quantidade de Operandos por Instrução:

Número de endereços de memória	Número máximo de operandos permitidos	Tipo de arquitetura	Exemplos
0	3	Carregamento-armazenamento/ registrador-registrador	Alpha, ARM, MIPS, PowerPC, SPARC, SuperH, TM32
1	2	Registrador-memória	IBM 360/370, Intel 80x86, Motorola 68000, TI TMS320C54x
2	2	Memória-memória	VAX (também possui formatos de três operandos)
3	3	Memória-memória	VAX (também possui formatos de dois operandos)

FIGURA A.3 Combinações típicas de operandos de memória e totais de operandos por instruções típicas da ULA com exemplos de computadores. Os computadores sem referência à memória por instrução da ULA são chamados de computadores de carregamento-armazenamento ou registrador-registrador. As instruções com vários operandos de memória por instruções típicas da ULA são chamadas de registrador-memória ou memória-memória, conforme tenham um ou mais de um operando de memória.

Instruções de ULA para Reg/Mem também podem ser implementadas com 1 operando em memória e 1 operando em registrador. **Qual é a implicação disso quanto:**

- À implementação do compilador?
- Ao tamanho (em bits) das instruções de diferentes classes?



Características das instruções de máquina

Resumo das estratégias quanto ao número de operandos:

Tabela 10.1 Utilização de endereços de instrução (instruções sem desvio)

Número de endereços	Representação simbólica	Interpretação
3	OP A, B, C	$A \leftarrow B \text{ OP } C$
2	OP A, B	$A \leftarrow A \text{ OP } B$
1	OP A	$AC \leftarrow AC \text{ OP } A$
0	OP	$T \leftarrow (T - 1) \text{ OP } T$

AC = acumulador

T = topo da pilha

$(T - 1)$ = segundo elemento da pilha

A, B, C = locais de memória ou registradores

Características das instruções de máquina

Projeto do conjunto de instruções: O projeto de um conjunto de instruções é muito complexo, pois afeta muitos aspectos do sistema de computador. Ele define muitas das funções realizadas pelo processador e, portanto, tem um efeito significativo sobre a implementação do processador. O conjunto de instruções é o meio de o programador controlar o processador

- **Repertório de operações:** quantas e quais operações oferecer, e que complexidade as operações deverão ter
- **Tipos de dados:** os diversos tipos de dados sobre os quais as operações são realizadas
- **Formato de instrução:** tamanho da instrução (em bits), número de endereços, tamanho dos diversos campos, e assim por diante
- **Registradores:** número de registradores do processador que podem ser referenciados pelas instruções e seu uso
- **Endereçamento:** o modo ou modos pelos quais o endereço de um operando é especificado

Tipos de operandos

As instruções de máquina operam sobre dados. As categorias gerais de dados mais importantes são:

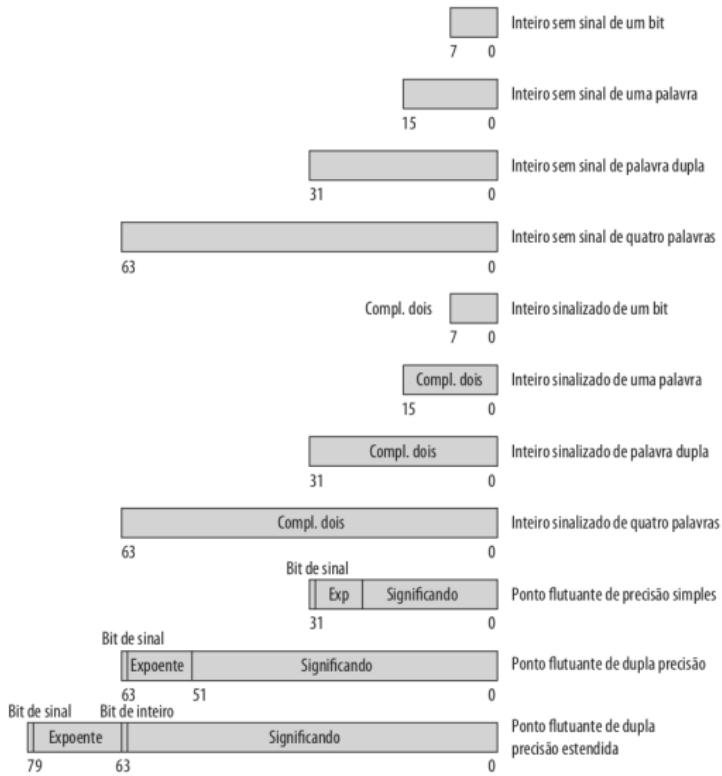
- **Endereços:** discutidos na Aula-11
- **Números:** inteiros binários; ponto flutuante; e decimais. Para decimais, pode-se usar o código BCD (do inglês binary coded decimal). Isso pode permitir evitar o *overhead* de conversão (binário-decimal). Alguns processadores oferecem operações nas representações decimais em BCD
- **Caracteres:** são armazenados no formato binário. Sistemas de conversão são definidos, como o ASCII (American Standard Code for Information Interchange) de 7 *bits* e o UNICODE¹, de 16 *bits*.
- **Dados lógicos:** arquiteturas podem implementar operações em *bits* em palavras. Neste caso, cada *bit* da sequência da palavra endereçável torna-se um unidade de dados lógicos.

Observe que os tipos de dados aqui descritos são definidos em linguagem de máquina, e não em linguagens de alto nível. Uma linguagem de alto nível pode especificar tipos de dados diferentes do oferecido na linguagem de máquina. Contudo, os tipos de operandos existentes na linguagem de baixo nível podem ser absorvidos e manipulados pelas linguagens de alto nível.

¹<https://pt.wikipedia.org/wiki/Unicode>

Tipos de operandos: Intel x86

Figura 10.4 Formatos de dados numéricicos x86



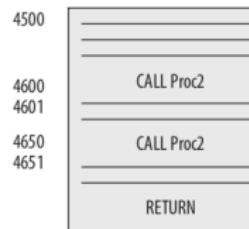
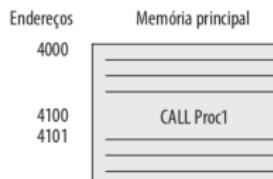
Tipos de operações

- **Transferência de dados:** são necessários (a) local dos operandos de origem e de destino, (b) tamanho dos dados e (c) o modo de endereçamento para cada operando
- **Aritmética:** geralmente, pelo menos as básicas são oferecidas (+, -, / e *)
- **Lógica:** oferece operações lógicas (or, and, not, xor e deslocamentos)
- **Conversão:** usadas para mudar o formato dos dados
- **E/S:** discutidas na Aula-07
- **Controle do sistema:** somente podem ser executadas enquanto o processador está em um certo estado privilegiado ou está executando um programa em uma área privilegiada especial da memória. Normalmente, são reservadas para o uso do sistema operacional.
- **Transferência de controle:** mudam a sequência de execução de instruções (saltos condicionais e incondicionais, instruções predicativas e chamadas de rotinas)

Tipos de operações: Chamada de procedimento

A modularidade de códigos em forma de procedimentos (métodos, funções) é um grande avanço na programação. Para isso, é necessário definir o endereço de chamada e o de retorno.

Figura 10.8 Procedimentos aninhados

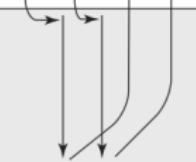
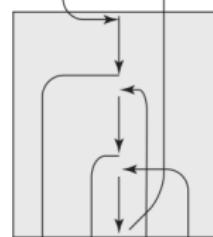
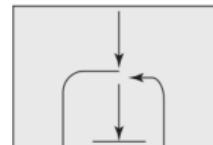


(a) Chamadas e retornos

Programa principal

Procedimento Proc1

Procedimento Proc2



(b) Sequência de execução

Tipos de operações: Chamada de procedimento

Existem três locais comuns para armazenar o endereço de retorno:

- registrador (limita chamadas de reentrantes, como recursivos)
- inicio do procedimento chamado (limita chamadas de reentrantes)
- topo da pilha (isso foi visto na Aula-07, Figura 7.7) (modo mais poderoso)

Também, é necessário passar os parâmetros:

- registradores (processador)
- armazenar os parâmetros na memória logo após a instrução CALL (no programa invocador)
- pilha (não limita o número de parâmetro e permite número flexível dentre os definidos)

Não deixe de ler no livro, no Capítulo 10:

- 10.3 Tipos de dados Intel x86 e do ARM
- 10.5 Tipos de operação Intel x86 e do ARM
- Apêndice 10A Pilhas
- Apêndice 10B Little, big e bi-endian

Tipos de operações: Chamada de procedimento

Existem três locais comuns para armazenar o endereço de retorno:

- registrador (limita chamadas de reentrantes, como recursivos)
- inicio do procedimento chamado (limita chamadas de reentrantes)
- topo da pilha (isso foi visto na Aula-07, Figura 7.7) (modo mais poderoso)

Também, é necessário passar os parâmetros:

- registradores (processador)
- armazenar os parâmetros na memória logo após a instrução CALL (no programa invocador)
- pilha (não limita o número de parâmetro e permite número flexível dentre os definidos)

Não deixe de ler no livro, no Capítulo 10:

- 10.3 Tipos de dados Intel x86 e do ARM
- 10.5 Tipos de operação Intel x86 e do ARM
- Apêndice 10A Pilhas
- Apêndice 10B Little, big e bi-endian

Agradecimentos

Agradecimentos Especiais:

Agradeço a toda a comunidade L^AT_EX.
Em especial a *Till Tantau* pelo *Beamer*.

<https://www.tcs.uni-luebeck.de/mitarbeiter/tantau/>

Desta forma, tornou-se possível a escrita deste material didático.

Exercícios:

Lista de exercícios divulgada no Moodle

