Arquitetura e Organização de Computadores

Capítulo 10

Conjuntos de instruções:
Características e funções

O que é um conjunto de instruções?

- A coleção completa de instruções que são entendidas por uma CPU.
- Código de máquina.
- Binário.
- Normalmente, representado por códigos em *assembly*.

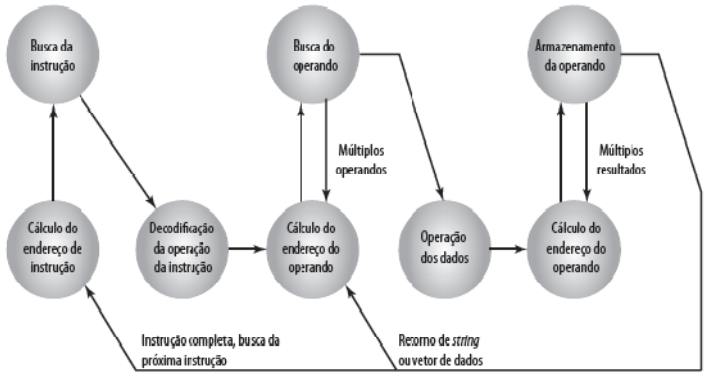
Elementos de uma instrução

- Código de operação (**Opcode**):
  —Faça isto.
- Referência a operando fonte:
  —Nisto.
- Referência a operando de destino:
  —Coloque a resposta aqui.
- Referência à próxima instrução:
  —Quando tiver feito isso, faça isto...

Para aonde foram todos os operandos?

- Muito tempo se passando….
- (Se você não entende, então é muito novo!)
- Memória principal (ou memória virtual ou cache).
- Registrador da CPU.
- Dispositivo de E/S.

Diagrama de estado do ciclo de instrução



Representação da instrução

- Em código de máquina, cada instrução tem um padrão de bits exclusivo.
- Para consumo humano (bem, para programadores), uma representação simbólica é utilizada.
  —P.e., ADD, SUB, LOAD.
- Operandos também podem ser representados desta maneira:
  —ADD A,B.

Formato de instrução simples

Tipos de instrução

- Processamento de dados.
- Armazenamento de dados (memória principal).
- Movimentação de dados (E/S).
- Controle de fluxo do programa.

Número de endereços (a)

- Instruções de 3 endereços:
  - —Operando 1, Operando 2, Resultado.
  - —a = b + c.
  - —Pode ser uma instrução for-next (normalmente implícita).
  - —Não é comum.
  - —Precisa de palavras muito longas para manter tudo.

Número de endereços (b)

- Instruções de 2 endereços:
  - —Um endereço servindo como operando e resultado.
  - —a = a + b.
  - —Reduz tamanho da instrução.
  - —Requer algum trabalho extra.
    - – Armazenamento temporário para manter alguns resultados.

Número de endereços (c)

- Instruções de 1 endereço:
  - —Segundo endereço implícito.
  - —Normalmente, um registrador (acumulador).
  - —Comum nas primeiras máquinas.

Número de endereços (d)

- 0 (zero) endereços:
  - —Todos os endereços implícitos.
  - —Usa uma pilha.
  - —p.e. push a.
  - — push b.
  - — add.
  - — pop c.
  - —c = a + b.

Quantos endereços

- Mais endereços:
  - —Instruções mais complexas (poderosas?).
  - —Mais registradores.
    - – Operações entre registradores são mais rápidas.
  - —**Menos instruções por programa**.
- Menos endereços:
  - —Instruções menos complexas (poderosas?).
  - —**Mais instruções por programa**.
  - —Busca/execução de instruções mais rápida.

Decisões de projeto do Conjunto de Instruções

- Repertório de operações:
  - —Quantas operações?
  - —O que elas podem fazer?
  - —Qual a complexidade delas?
- Tipos de dados.
- Formatos de instrução:
  - —Tamanho do campo de código de operação.
  - —Número de endereços.

- Registradores:
  - —Número de registradores da CPU disponíveis.
  - —Quais operações podem ser realizadas sobre quais registradores?
- Modos de endereçamento (mais adiante…).

- RISC v CISC.

Tipos de operando

- Endereços.
- Números:
  - —Inteiro/ponto flutuante.
- Caracteres:
  - —ASCII etc.
- Dados lógicos:
  - —Bits ou flags

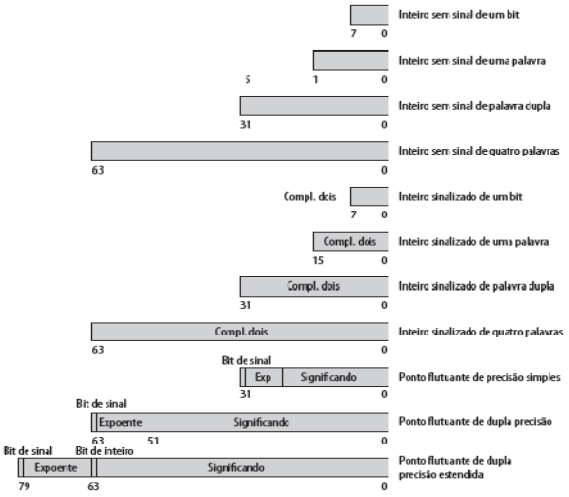
Tipos de dados do x86

- Byte de 8 bits.
- Word de 16 bits.
- Palavras duplas de 32 bits (***double word***).
- Quatro palavras de 64 bits (***quad word***).
- Quatro palavras duplas de 128 bits.
- Endereçamento por unidade de 8 bits.
- Palavras não precisam alinhar em endereço com número par.
- Dados acessados pelo barramento de 32 bits em unidades de palavras duplas lidas em endereços divisíveis por 4.
- **Little endian** (armazena o byte + significativo no endereço numérico + alto).

Tipos de dados SIMD (*Single Instruction Multiple Data*)

- Tipos inteiros:
  - — Interpretados como campo de bit ou inteiro.
- Byte agrupado e inteiro de byte agrupado:
  - — Bytes agrupados em *quadword* de 64 bits ou *double quadword* de 128 bits.
- Palavra agrupada e inteiro de palavra agrupada:
  - — Palavras de 16 bits agrupadas em *quadword* de 64 bits ou *double quadword* de 128 bits.
- *Doubleword* agrupado e inteiro de *doubleword* agrupado.
  - — *Doubleword* de 32 bits agrupados em *quadword* de 64 bits ou *double quadword* de 128 bits.
- *Quadword* agrupado e inteiro de *quadword* agrupado:
  - — Duas *quadwords* de 64 bits agrupadas em *double quadword* de 128 bits.
- Ponto flutuante de precisão simples agrupado e ponto flutuante de precisão dupla agrupada:
  - — Quatro valores de ponto flutuante de 32 bits ou dois valores de ponto flutuante de 64 bits em uma *double quadword* de 128 bits.

Formatos de dados numéricos do x86



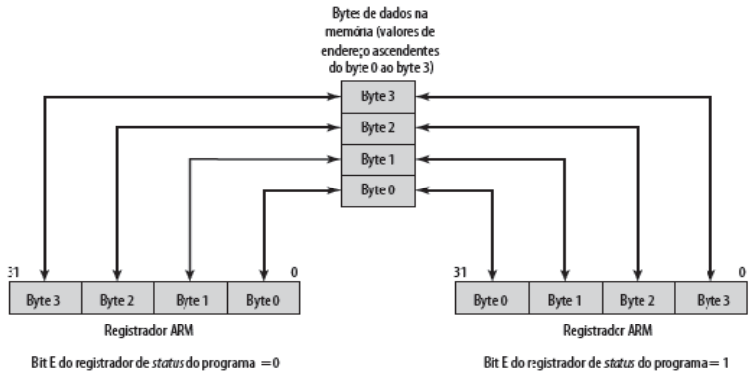
Tipos de dados do ARM

- 8 (byte), 16 (meia-palavra), 32 (palavra) bits.
- Acessos de meia-palavra e palavra devem ser alinhados por palavra.
- Alternativas de acesso não alinhado.
  - — Default:
    - – Tratado como truncado.
    - – Bits[1:0] tratado como zero para word.
    - – Bit[0] tratado como zero para meia-palavra.
    - – Carrega instruções de única palavra, gira para direita dados alinhados por palavra transferidos por endereço não alinhado por palavra com um, dois ou três bytes.
    - – Verificação de alinhamento.
  - — Sinal de abortar dados indica falta de alinhamento para tentar acesso desalinhado.
  - — Acesso desalinhado.
  - — Processador usa um ou mais acessos à memória para gerar transferência de bytes adjacentes de forma transparente ao programador.

- Interpretação de inteiro desalinhado aceita para todos os tipos.
- Interpretação de inteiro com sinal de complemento a dois aceita para todos os tipos.
- Maioria das implementações não oferece hardware de ponto flutuante.
  - —Economiza energia e superfície.
  - —Aritmética de ponto flutuante implementada no software.
  - —Coprocessador de ponto flutuante.
  - —Tipos de dados de ponto flutuante IEEE 754 de precisão simples e dupla.

Suporte a endian no ARM

- Bit E no registrador de controle do sistema.
- Sob controle do programa.



Tipos de operação no Conjunto de Instruções

- Transferência de dados.
- Aritmética.
- Lógica.
- Conversão.
- E/S.
- Controle do sistema.
- Transferência de controle.

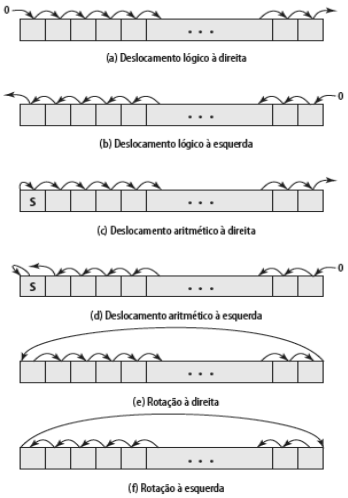
Transferência de dados

- Especificam:
  - —Origem.
  - —Destino.
  - —Quantidade de dados.
- Podem ser instruções diferentes para diferentes movimentações.
  - —P.e., IBM 370.
- Ou uma instrução e diferentes endereços.
  - —P.e., VAX.

Aritmética

- Adição, Subtração, Multiplicação, Divisão.
- Inteiro com sinal.
- Ponto flutuante?
- Pode incluir:
  - —Incremento (a++).
  - —Decremento (a--).
  - —Negação (-a).

Operações de deslocamento e rotação



Lógica

- Operações bit a bit.
- AND, OR, NOT.

Conversão

- P.e., binário para decimal.

Entrada/saída

- Podem ser instruções específicas.
- Pode ser feita usando instruções de movimentação de dados (mapeadas na memória).
- Pode ser feita por um controlador separado (DMA).

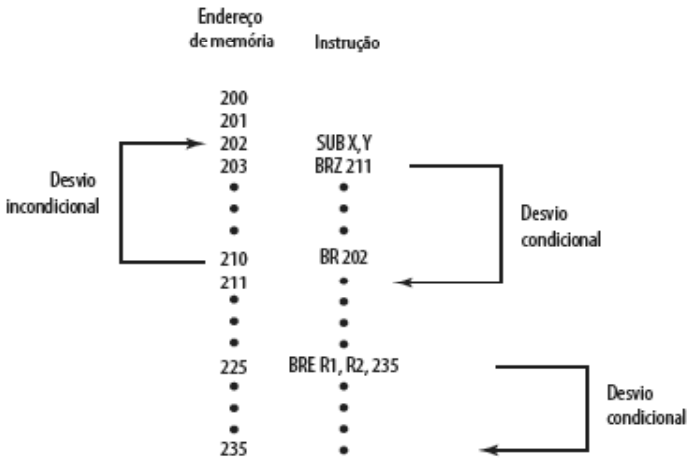
Controle do sistema

- Instruções privilegiadas.
- CPU precisa estar em estado específico:
  —Modo kernel.
- Para uso dos sistemas operacionais.

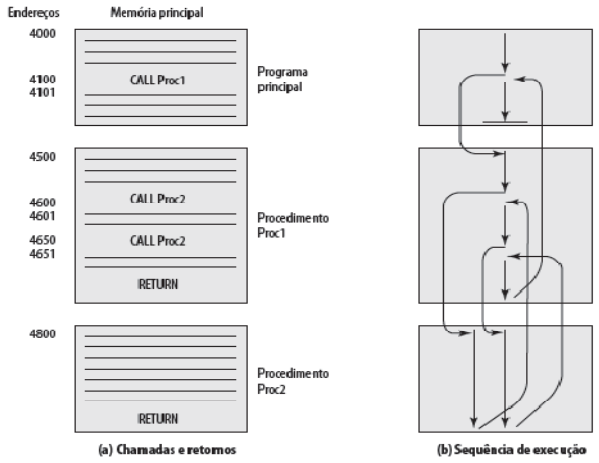
Transferência de controle

- Desvio:
  - —P.e., desvio para x se resultado for zero.
- Salto:
  - —P.e., incrementa e salta se for zero.
  - —ISZ Registrador 1.
  - —Desvia xxxx.
  - —ADD A.
- Chamada de sub-rotina:
  - —C.f. chamada de interrupção.

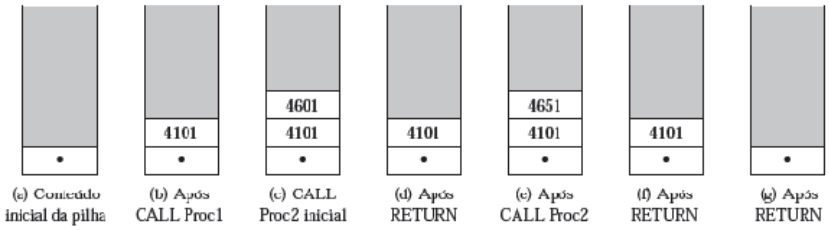
Instrução de desvio



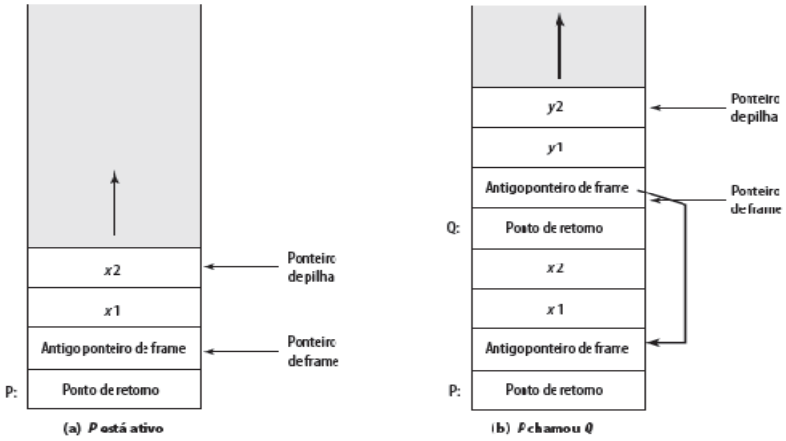
Chamadas de procedimento aninhadas



Uso da pilha



Crescimento do frame de pilha usando procedimentos de exemplo P e Q



Exercício para o leitor

- Descubra sobre o conjunto de instruções do Pentium e ARM.
- Comece com Stallings.
- Visite *Web sites*.

Ordem do byte
(Uma repartição dos chips?)

- Em que ordem lemos números que ocupam mais de um byte?
- P.e., números em hexa para facilitar a leitura.
- 12345678 pode ser armazenado em 4 locais de 8 bits, da forma a seguir.

Ordem do byte (exemplo)
Supondo valor de 32 bits = 12345678

Endereço	Valor (1)	Valor (2)
184	12	78
185	34	56
186	56	34
186	78	12

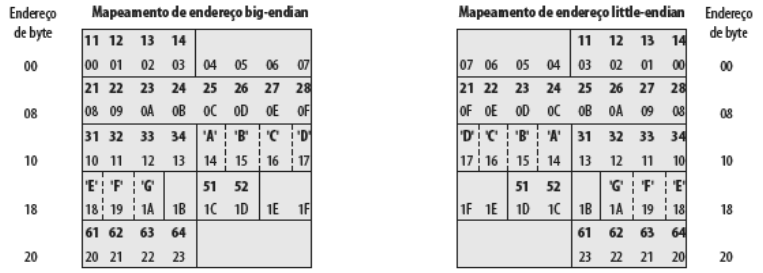
- Ou seja, ler de cima para baixo ou de baixo para cima?

Nomes de ordem de byte

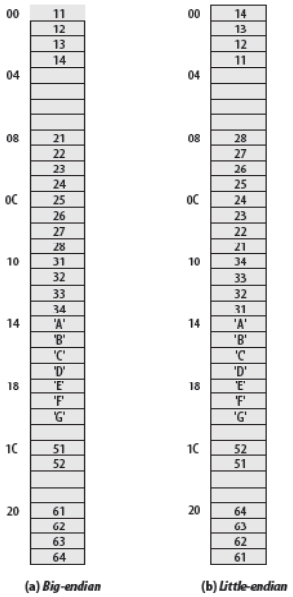
- O problema se chama **Endian**.
- O mapeamento à esquerda tem o byte mais significativo no endereço numérico mais baixo.
- Isso é chamado de **big-endian**.
- O mapeamento à direita tem o byte menos significativo no endereço numérico mais baixo.
- Isso é chamado de **little-endian**.

Exemplo de estrutura de dados em C

```
struct{
   int      a;     //0x1112_1314                    word
 int        pad;   //
   double b;       //0x2122_2324_2526_2728          doubleword
   char*    c;     //0x3132_3334                    word
   char     d[7];  //'A'.'B','C','D','E','F','G'    byte array
   short    e;     //0x5152                         halfword
   int      f;     //0x6162_6364                    word
} s;
```



Visão alternativa do mapa de memória



Padrão... Que padrão?

- Pentium (x86), VAX são **little-endian**.
- IBM 370, Motorola 680x0 (Mac) e a maioria dos RISCs são **big-endian**.
- Internet é **big-endian**.
  - —Torna a escrita de programas para Internet no PC mais desajeitada!
  - —WinSock oferece funções htoi e itoh (Host to Internet & Internet to Host) para conversão.