Linguagens de Montagem
Introdução ao processador ARM

Ricardo Anido
Instituto de Computação
Unicamp

Registradores

- O ARM tem 16 registradores, que são identificados por r0, r1, r2, etc.
- Na versão que vamos utilizar, os registradores são de 32 bits.
- Alguns registradores são reconhecidos por mais de um nome; esse outro nome em geral especifica um uso comum do registrador.
- O registrador r15 é o *contador (ou apontador) de programa*, e é também reconhecido pelo nome pc. Ele armazena o endereço da próxima instrução a ser executada (e é automaticamente atualizado a cada instrução executada).

Instruções de transferência de dados

- Permitem a transferência de dados entre dois registradores do processador ou entre a memória e um registrador.

Copia valor de registrador para registrador

Copia o valor de um registrador (chamado registrador fonte) para um outro registrador (chamado registrador destino).
O comando em linguagem de montagem é "MOV".

```
@ exemplos de instrução mov
   mov   r1,r5    @ copia valor de r5 para r1
   mov   r2,r1    @ copia valor de r1 para r2
```

Carrega registrador com valor da memória

- Carrega um registrador com valor da memória.
- Diversas formas, com diferentes *modos de endereçamento*.
- O modo de endereçamento de uma instrução especifica a forma como o endereço do operando na memória é calculado pelo processador.
- Vamos inicialmente ver três modos de endereçamento: imediato, direto e indireto por registrador.

Endereçamento imediato

- no endereçamento imediato o valor do operando faz parte do próprio código da instrução.
- Na instrução *Carrega registrador com endereçamento imediato* o registrador destino é carregado com o valor do operando, dado com endereçamento imediato.
- o valor carregado no registrador é sempre o mesmo!
- utilizada quando se deseja carregar um valor constante no registrador destino.

Carrega registrador com endereçamento imediato

Copia um valor constante para um registrador (chamado registrador destino). No ARM a constante a ser carregada é codificada em 12 bits, portanto nem toda constante pode ser carregada utilizando esta instrução. Veremos mais adiante como carregar constantes de mais de 12 bits.
Na linguagem de montagem do ARM ambém utiliza o comando "MOV".

```
@ exemplos de instrução mov para carregar constantes
   .equ MAX,255

   mov   r1,#100   @ carrega o valor 100 (decimal) no registrador 1
   mov   r2,#0x100 @ carrega o valor 100 (hexadecimal) no registrador 2
   mov   r2,#MAX   @ carrega o valor 255 (decimal) no registrador 3
```

Endereçamento direto

- No endereçamento direto, a a instrução contém o *endereço* da posição de memória cujo valor se deseja acessar.
- A instrução *Carrega registrador com endereçamento direto* (LDR) carrega um registrador com o valor de uma posição de memória, cujo endereço é especificado na instrução.
- no ARM, o valor do endereço do operando é "relativo" ao valor do registrador pc. Ou seja, para determinar o endereço efetivo do operando, o processador adiciona o valor do endereço codificado na instrução com o valor corrente do registrador pc. O valor obtido é o endereço da memória cujo conteúdo o processador copiará no registrador destino.

Carrega registrador com endereçamento direto

Para executar uma instrução LDR com endereçamento direto o processador ARM faz dois acessos à memória:

- um acesso no endereço apontado pelo registrador `pc`, para a busca da instrução;
- o processador adiciona o valor corrente do registrador ao endereço relativo codificado na instrução, obtendo o "endereço efetivo" do operando.
- o processador então faz outro acesso à memória para a busca do valor operando no endereço efetivo calculado. O valor obtido nesse último acesso é armazenado no registrador *rd*.

Carrega registrador com endereçamento direto

```
@ exemplos de instrução ldr com endereçamento direto

      ldr   r9,cont      @ carrega conteúdo do
                         @ endereço cont
      ldr   r10,cont+4   @ carrega conteúdo do
                         @ endereço cont+4
      ...

cont:                    @ uma variável
      .word -1           @ associada
      .word 0x80000000   @ ao endereço 0x2000
```

Endereçamento indireto por registrador

- endereço do operando é dado em um registrador, ao invés de ser um valor constante, codificado na instrução, como no endereçamento direto.
- Em linguagem de montagem usa o mesmo comando LDR já utilizado anteriormente, mas indicaremos o modo de endereçamento distinto pela grafia do operando fonte.

Carrega registrador com endereçamento indireto por registrador

```
@ exemplo instrução ldr (indir. por registrador)
      ldr  r6, [r5]       @ instrução ldr
```

Armazena registrador

- A instrução *Armazena registrador* (STR, do inglês *store*, armazenar), efetua a operação inversa à operação LDR, armazenando na memória o valor de um registrador.
- Da mesma forma que a instrução LDR temos duas formas: com endereçamento direto e indireto por registrador.

Armazena registrador

```
@ exemplos de instrução st

      str  var,r7        @ armazena registrador
                         @ r7 na variável var
      str  r3,[r4]       @ r4 tem endereço
                         @ onde queremos
                         @ armazenar r3
       .org 0x1000
var:                     @ uma variável montada
       .skip 4           @ no endereço 0x1000
```

Carrega (ou Armazena) bytes para (de) registrador

As instruções de carga e armazenamento podem também carregar/armazenar bytes ao invés de palavras. Em linguagem de montagem do ARM acrescentamos o sufixo "B" ao comando (LDRB e STRB)

Armazena byte de registrador

```
@ exemplo de instruções ldrb e strb

    strb  r4,carac    @ armazena byte menos significativo de r4
                      @ no endereço de rótulo carac
    ldrb  r2,[r5]     @ carrega byte menos significativo da palavra
                      @ de memória cujo endereço é o valor de r5

     .org 0x2000
carac:                @ uma variável montada
     .ascii "ABCDabc" @ no endereço 0x2000
                      @ que representa uma
                      @ sequência de oito
                      @ caracteres
```

Usos de modos de endereçamento

- Os diferentes modos de endereçamento fornecidos por um processador podem ser utilizados para implementar diferentes conceitos em linguagens de programação.
- O modo de endereçamento direto pode ser usado para carregar e armazenar valores de variáveis que podem ser acessadas diretamente, através dos rótulos associados às variáveis.

Exemplo

```
// Trecho de programa em C
#define TAMANHO 256

int a,b;
char c,d=1;
...

a = TAMANHO;
b = a;
c = 'a';
```

Exemplo

```
@ definição de uma constante
    .equ TAMANHO,256

@ reserva espaço para as variáveis
    .org  0x400
a:  .skip 4                 @ variável inteira, quatro bytes
b:  .skip 4                 @ variável inteira, quatro bytes
c:  .skip 1                 @ variável char, reserva um byte
d:  .byte 1                 @ variável char, inicializada, valor é 1

    .org  0x800
    mov   r0,#TAMANHO       @ r0 usado como auxiliar
    str   r0,a              @ armazena valor na variável a
    str   r0,b              @ armazena valor na variável b
    mov   r1,#0x61          @ r1 usado como auxiliar; note
                            @ que r0 poderia ser reutilizado
                            @ ao invés de utilizar r1
    strb   r1,c             @ armazena caractere na variável c
```

Usos de modos de endereçamento

- O modo de endereçamento indireto por registrador pode ser usado para implementar o conceito de apontadores em linguagens de alto nível: carregamos o endereço de uma variável em um registrador, e usamos esse registrador para acessar essa variável.
- Se a variável é uma coleção de valores, por exemplo um vetor de inteiros, podemos *percorrer* os elementos da variável incrementando ou decrementando o registrador para acessar diferentes elementos.

Exemplo

```
// Trecho de programa em C

int x=1,vet[100],*p;
...

p = &vet[0];
*p = x;
```

Exemplo

```
@ reserva espaço para as variáveis
.org  0x400

x:    .word 1               @ variável inteira, valor 1
vet:  .skip 4*100           @ variável vetor, inteira, 100 elementos
p:    .skip 4               @ variável apontador, quatro bytes

ldr   r0,=vet     @ como visto em aula, esta construção da linguagem do
                  @ arm armazena em r0 endereço do rótulo vet
                  @ r0 vai ser usado como apontador p, tem endereço de vet
str   r0,p        @ armazena valor na variável p
ldr   r1,x        @ r1 tem valor da variável x
str   r1,[r0]     @ armazena valor de x no elemento 0 de vet
```