Arquitetura de Computadores

ISA – Instruções de processamento de dados

Bib: A – secções 6.1, 6.2 e 6.3.1

João Pedro Patriarca (jpatri@cc.isel.ipl.pt), Gabinete F.0.23 do edifício F

ISEL, ADEETC, LEIC

Instruções de processamento de dados

- Dividem-se em dois grandes grupos
  - Aritméticas
  - Lógicas

- Todas as instruções de processamento de dados produzem, para além do resultado da operação, um conjunto de *flags* que são guardadas no registo CPSR (*Current Program Status Register*)

- Operações binárias envolvem três operandos: dois, para operandos; um, para resultado
  - Operando A ∈ {R0, R7}
  - Operando B ∈ {R0, R15}
  - Resultado ∈ {R0, R15}

- Conjunto de instruções e respetivas codificações: manual de consulta rápida das instruções do P16

Desafios

1. Adicionar A+B, ambos de 32 bits. O operando A está guardado no par R0 (*low*) e R1 (*high*). O operando B está guardado no par R2 (*low*) e R3 (*high*). O resultado fica no par R0 (*low*) e R1 (*high*)

2. Complementar o valor presente em R0 utilizando exclusivamente instruções de processamento de dados. Deixar o resultado em R0.

3. Dividir por 16, nos naturais, o valor presente em R0. Deixar o resultado em R0.

4. Dividir por 16, nos relativos, o valor presente em R0. Deixar o resultado em R0.

5. Multiplicar por 8 o valor presente em R0. Deixar o resultado em R0.

6. Multiplicar o valor presente em R0 por 25. Deixar o resultado em R0.

Desafio 1

- Adicionar A+B, ambos de 32 bits. O operando A está guardado no par R0 (*low*) e R1 (*high*). O operando B está guardado no par R2 (*low*) e R3 (*high*). O resultado fica no par R0 (*low*) e R1 (*high*)

```
/*
Comentários no assembly do P16:
1. /* Comentário de bloco */
2. ; Comentário de linha
uint32_t A;   // Variável A mapeada no par de registos R1:R0
uint32_t B;   // Variável B mapeada no par de registos R3:R2
A = A + B;
*/
add r0, r0, r2 ; adiciona os 16 bits de menor peso
adc r1, r1, r3 ; adiciona os 16 bits de maior peso considerando o arrasto produzido pela
               ; adição anterior
```

Desafio 2

- Complementar o valor presente em R0 utilizando exclusivamente instruções de processamento de dados. Deixar o resultado em R0.

```
/*
int16_t A;   // Variável A mapeada no registo R0
A = -A;
*/
eor r1, r1, r1 ; coloca o valor 0 no registo R1
sub r0, r1, r0 ; subtrai a 0 o valor da variável A
```

Desafio 3

- Dividir por 16, nos naturais, o valor presente em R0. Deixar o resultado em R0.

```
/*
uint16_t A;   // Variável A mapeada no registo R0
A = A / 16;
*/
shr r0, r0, 4 ; desloca o valor da variável A 4 bits para a direita introduzindo sempre 0s
```

Desafio 4

- Dividir por 16, nos relativos, o valor presente em R0. Deixar o resultado em R0.

```
/*
int16_t A;   // Variável A mapeada no registo R0
A = A / 16;
*/
asr r0, r0, 4 ; desloca o valor da variável A 4 bits para a direita introduzindo o valor do
              ; bit de sinal
```

Desafio 5

- Multiplicar por 8 o valor presente em R0. Deixar o resultado em R0.

```
/*
uint16_t A;   // Variável A mapeada no registo R0
A = A * 8;
*/
shl r0, r0, 3 ; desloca o valor da variável A 3 bits para a esquerda
```

Desafio 6

- Multiplicar o valor presente em R0 por 25. Deixar o resultado em R0.

```
/*
uint16_t A;   // Variável A mapeada no registo R0
A = A * 25;   // Multiplicar A por 25 é equivalente a fazer A * (16 + 8 + 1), ou seja
              // 16 * A + 8 * A + A
*/
shl r1, r0, 4  ; multiplica A por 16
shl r2, r0, 3  ; multiplica A por 8
add r1, r1, r2 ; adiciona A * 16 com A * 8
add r0, r1, r0 ; produz resultado final somando ao resultado parcial o valor da variável A
```