Arquitetura AArch64

Nos exercícios seguintes utilizar a seguinte abordagem:

1. Escrever a solução do exercício em *assembly* na forma de uma sub-rotina. Ter em consideração a convenção de chamada de sub-rotinas – a passagem de argumentos e a devolução do resultado faz-se nos registos `X0` a `X7`, por ordem.
2. Na função `main` do programa em C declarar os dados necessários para testar a sub-rotina, chamar a sub-rotina em causa e imprimir o resultado da execução. Ter em consideração a compatibilidade do tipo de dados das variáveis e os registos a utilizar (`Xn` ou `Wn`).
3. Caso o programa não execute como esperado, fazer *debug* no modo passo a passo e seguir atentamente o conteúdo dos registos e de memória resultante das instruções e respetivo fluxo de execução.

1. Pretende-se calcular a soma dos `N` elementos de um vetor (*array*) `V`.

a) Implementar a sub-rotina `SOMA_V`, em *assembly* AArch64, considerando o seguinte programa em linguagem C:

```c
#include <stdio.h>
#include <stdlib.h>
extern int SOMA_V(int *a, int n);
int main(void)
{
    int dim = 5;
    int v[] = {3, -1, 8, 0, -3};   // Para testar : 32 bits
    int res;
    res = SOMA_V(v, dim);
    printf("Soma dos elementos = %d\n", res);
    return EXIT_SUCCESS;
}
```

b) Descrever que alterações seriam necessárias se o vetor tivesse valores de 64 bits.

Mudar todos os registo tipo W para tipo X

2. Escrever e testar programas que usem uma sub-rotina em *assembly* para calcular:

a) o valor máximo de um vetor com dados do tipo `doubleword`.

b) o valor mínimo de um vetor com dados do tipo `word`.

c) o valor máximo de um vetor com dados do tipo `halfword`.

d) o valor médio (inteiro) de um vetor com dados de 64 bits.

e) o número de valores de um vetor que pertencem ao intervalo $[a; b]$.

3. Escrever e testar programas que usem uma sub-rotina em *assembly* para calcular:

a) Contar o número de bits a 1 na representação binária de um valor do tipo `doubleword`.

b) Determinar o número de bits iguais em posições correspondentes de dois valores do tipo `word`.

c) Contar quantos números pares existem num vetor de inteiros.

d) Determinar quantas vezes a sequência '10110' aparece na representação binária de um valor do tipo `word`:

i. sem sobreposições.

ii. com sobreposições.

4. Pretende-se escrever programas que executam tarefas envolvendo uma cadeia de carateres terminada por zero (`'\0'`).

- **a)** Determinar o comprimento de uma cadeia de carateres.
- **b)** Determinar o número de ocorrências de um caráter numa cadeia de carateres.
- **c)** Determinar o número de vogais de uma cadeia de carateres.
- **d)** Determinar o número de letras maiúsculas de uma cadeia de carateres.
- **e)** Palíndromo é uma palavra, grupo de palavras ou verso em que o sentido é o mesmo, quer se leia da esquerda para a direita quer da direita para a esquerda (ex.: "ANOTARAM A DATA DA MARATONA").
  Implementar um programa que determine se uma cadeia de carateres é palíndromo. Assumir que a cadeia de carateres não inclui espaços nem sinais de pontuação e que é indiferente o uso de letras maiúsculas e minúsculas. A sub-rotina devolve 1 em caso afirmativo ou 0 no caso contrário.
- **f)** Contar quantas palavras tem uma cadeia de carateres, assumindo que há um único espaço entre palavras consecutivas.

5. Implementar e testar programas que realizem as seguintes tarefas:

- **a)** Copiar um vetor com valores (sem sinal) do tipo `byte` para um vetor com valores do tipo `doubleword`.
- **b)** Copiar um vetor com valores (com sinal) do tipo `word` para um vetor com valores do tipo `doubleword`.

6. Escrever e testar sub-rotinas considerando os respetivos protótipos da função a invocar em C.

- **a)** Determinar a posição do bit 1 mais significativo da representação binária de um número.

```
int POS1msb(long int n);
```

- **b)** Verificar se uma cadeia de 8 carateres (tamanho de uma *doubleword*) é palíndromo. A resposta será 1 em caso afirmativo e 0 no caso contrário.

```
int PAL8(char *s);
```

- **c)** Verificar se a representação binária de um número é capicua (número palíndromo). A resposta será 1 em caso afirmativo e 0 no caso contrário.

```
int NCAP(int n);
```

7. Implementar as instruções seguintes com base em instruções de manipulação ou extração de bits.

- **a)** `LSL X10, X12, 8`
- **b)** `ASR X10, X12, 8`
- **c)** `ROR W13, W14, 3`

8. Determinar o valor de `W0` após a execução de cada fragmento de código.

- **a)**

```
mov   W0, 0x66666666
mov   W1, 0xF000000F
and   W0, W0, W1
```

```
eor   W0, W0, W1
orr   W0, W0, \#0x66666666
```

b)

```
mov   W0, 0x0000BEEF
adds  W0, W0, 0x00008000
mov   W1, 0x00003EEE
sbc   W0, W0, W1
adc   W0, W0, W0
```

9. Considerar que o valor inicial de `W0` é `0x12345678`. Determinar o valor de `W1` (ou `X1`) após a execução de cada fragmento de código.

a)

```
mov   W1, 0xABCD0000
ubfx  W2, W0, 24, 8
bfi   W1, W2, 16, 8
```

b)

```
rev   W1, W0
and   W1, W1, W1, ASR 16
rev   W1, W1
sub   W1, W0, W1
```

c)

```
eon   X1, X1, X1
add   X1, X1, W0, SXTB 4
```

10. Identificar o significado do valor de `X0` após a execução de cada fragmento de código.

a)

```
cmp   X0, 0
cneg  X0, X0, LT
```

b)

```
cmp   X1, X2
csel  X0, X1, X2, GT
cmp   X0, X3
csel  X0, X0, X3, GT
```

11. Tendo em consideração a representação binária de um número inteiro, implementar uma sub-rotina que determine o número mínimo de bits com que pode ser representado.

12. Implementar e testar programas que implementem as seguintes operações com vetores de números inteiros:

- **a)** Adição de vetores – assumir que a soma é representável com o mesmo número de bits dos operandos, ou seja, assumir que não ocorre *overflow*.
- **b)** Adição de vetores – se ocorrer *overflow* ao somar dois valores usar o maior ou o menor valor representável, conforme essa soma seja, respetivamente, positiva ou negativa.
- **c)** Multiplicação de um vetor por um inteiro – assumir que o vetor original é substituído pelo vetor resultante (assumir a não ocorrência de *overflow*).

d) Produto interno – assumir que não ocorre *overflow* durante o cálculo.

e) Produto interno – caso ocorra *overflow*, a sub-rotina deve assinalar essa situação retornando o maior inteiro representável (assumir que este valor nunca decorre do cálculo do produto interno).

Fim do enunciado