AArch64: Programação em linguagem *assembly*

Nos exercícios cuja solução requer código "assembly" deve ser usada 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. Em cada alínea existe apenas uma resposta certa.

a) Considere que `W3=0x8ABC0DEF` antes de executar o código seguinte. Indique o valor final de `W3`.

```
str     W3, [X0]
ldrsb   W3, [X0]
```

- ☐ `0x8ABC0DEF`
- ☒ `0x000000EF`
- ☐ `0xFFFFFFEF`
- ☐ `00000DEF`

b) Qual das seguintes instruções dá erro de compilação?

- ☐ `SUB X2,X10,X2,ASR #2`
- ☐ `SMULH X0,X0,X0`
- ☐ `ANDS W1,W2,W3`
- ☒ `LDR W1,[W0,#4]` — `X0`

c) Considere o seguinte programa composto por código C e *assembly* `AArch64`.

```c
unsigned long int SUBR_T2(char *p);

int main(void) {
char s[] = "30 de Abril";
unsigned long int res;

res = SUBR_T2(s);
printf("0x%x\n", res); //Imprime em
    hexadecimal
return EXIT_SUCCESS;}
```

```
.text
.global SUBR_T2
.type SUBR_T2, "function"

SUBR_T2: LDRB W1, [X0]
        CBZ   W1, FIM
        ADD   X0, X0, #1
        B     SUBR_T2
FIM:    RET
```

Após execução é escrito no ecrã `0x00000000FEFFFFD3`.
Qual o endereço de memória ocupado pelo primeiro elemento ('3') da cadeia de carateres s?

- ☐ `0x00000000FEFFFFD2`
- ☐ `0x00000000FEFFFFC7`
- ☒ `0x00000000FEFFFFC8`
- ☐ `0x00000000FEFFFFBC`

d) Supor que `x0=0x10000000` e que a tabela representa o conteúdo da memória após executar a instrução `str w9,[x0]`. Qual o conteúdo inicial de `w9` ?

Endereço (hex)	Conteúdo (hex)
10000003	0F
10000002	31
10000001	60
10000000	0C

☐ 0x0F31600C

☐ 0xC00613F0

☐ 0xF01306C0

☐ 0x0C60310F

e) Considere que `SP=0x0805034A0` e que após execução das instruções seguintes o estado da pilha é o indicado na tabela.

```
smaddl   X1, W1, W1, X1
str      X1, [SP, -16]!
```

Endereço (hex.)	Conteúdo (hex.)
0805034A8	0FF
0805034A0	001
080503498	000
080503490	00C

Nestas circunstâncias, pode afirmar-se que o valor inicial de `X1` é:

☐ 1 ☐ -1 ☐ 0 ☐ 3

f) Considere a declaração `extern int ism(int x, int y);` e respetiva rotina em *assembly* `AArch64`.

```
ism: sdiv   w2,w0,w1
     msub   w0,w1,w2,w0
     cmp    w0,#0
     cset   w0,eq
     ret
```

W2 = 4
W0 é o resto

Qual o valor de `m = ism(34,7)` ?

☐ 1 ☐ 4 ☒ 0 ☐ 6

g) Considere a declaração `extern int vsum(int *a, int n);` e respetiva rotina em *assembly* `AArch64` que pretende calcular a soma dos elementos de um vetor.

```
vsum: eor    x10,x10,x10   x10 = 0
nxt:  cbz    x1,stop
      ldrsw  x9,[x0],#4    w = 32bits
      add    x10,x10,x9
      sub    x1,x1,#1
      b      nxt
stop: mov  x0,x10
      ret
```

Qual das afirmações é verdadeira?

☒ A rotina está correta.

☐ A rotina está errada pois retorna $n \times v[1]$.

☐ A rotina está errada pois retorna $n \times v[0]$.

☐ A rotina está errada pois retorna $(n-1) \times v[1]$.

2. Considerar a execução do fragmento de código indicado a seguir.

```
L1:   cbz   w1, L2
      ldr   w2, [x6], 4
      sub   w1, w1, 1
      eor   w0, w0, w2
L2:   b     L1
L2:   ...
```

Inicialmente, `w0=0, w1=12`. Quantas instruções são executadas?

12 * 5 + 1

3. Considere o seguinte programa composto por dois ficheiros.
Ficheiro em linguagem C (`main.c`):

```
extern int SUBROT1(int *a, int d);
int main()
{
  int n = 0, tam = 5;
  int seq[] = {1, 3, 6, 1, 9};
  n = SUBROT1(seq, tam);
  printf("%d\n", n);
  return EXIT_SUCCESS;
}
```

Ficheiro em linguagem *assembly*:

```
SUBROT1:
STP   X29,X30,[SP, #-16]!  //<1>
MOV   X29,SP

MOV   W10,0
LDR   W12, [X0]  word 32 bits = 4bytes
ADD   X0, X0, #4
SUB   X1, X1, #1
MOV   W11, #1
CICLO:
CBZ   X1, FIM
LDR   W13, [X0]  word
CMP   W13, W12
B.LE  FSC                  //<2>
ADD   W11, W11, #1
B     PROX
```

X29 - Stack pointer
X30 - return address

```
                Próximo valor é menor
FSC:
CMP   W10, W11
B.GE  PROX
MOV   W10, W11
MOV   W11, #1

PROX:
MOV   W12, W13
ADD   X0, X0, #4
SUB   X1, X1, #1  X1 = X1 - 1
B     CICLO

FIM:   MOV   W0, W10
CMP   W10, W11
B.GE  TERMINAR
MOV   W0, W11
TERMINAR:
LDP   X29, X30, [SP], #16
RET
```

a) Quanta informação, em número de palavras (*words*), é lida de memória ?

☐ 9 ☒ 5 ☐ 6 ☐ 7

b) Assumindo que o valor inicial do registo `SP` é `0xE0`, indique em que endereço da pilha se encontra armazenado o registo `X30` após a execução da instrução assinalada com <1>.

☐ 0xE0 ☐ 0xE8 ☒ 0xD8 ☐ 0xD0

0xE0 - 8 = 0xD8

c) Para os valores de invocação indicados em `main.c`, quantas vezes é tomado o salto assinalado com <2>?

☐ 2 ☒ 1 ☐ 3 ☐ 4

d) Para os valores indicados no programa, qual é o resultado apresentado no ecrã?

☐ 5 ☐ 2 ☒ 3 ☐ 9

4. Para este exercício usar a ferramenta de correção automática de `https://aoco.f.e.up.pt`.

Escrever uma sub-rotina que substitui por 0 (zero) os elementos de uma sequência cujo valor absoluto é maior que o valor positivo X, devolvendo o número de elementos modificados. Os elementos da sequência são do tipo `signed word`. A sub-rotina deve ter o nome `CheckABS` e aceitar os seguintes argumentos pela ordem indicada:

1. valor `X` (do tipo `unsigned word`);
2. número de elementos da sequência (do tipo `unsigned word`);
3. endereço-base da sequência.

Para efeitos de teste, pode usar-se o seguinte código C:

```c
#include <stdio.h>
#include <stdlib.h>
extern int CheckABS( int numx, int tam, int *seq);
int main(void)
{
    int x = 20, tamanho = 8;
    int ve[] = {7, -8, -23, 56, 20, -10, 0, 40};
    int res;

    res = CheckABS(x, tamanho, ve);
    printf("Foram modificados %d elementos da sequencia ve[]", res);
    // para o exemplo fornecido o output deve indicar que foram alterados 3 elementos
    // no final da execução a sequência ve[] = 7, -8, 0, 0, 20, -10, 0, 0 (podem verificar
    //o seu conteúdo durante a execução em modo debug)
    return EXIT_SUCCESS;
}
```

Usar o ficheiro `CheckABS.s` disponível no `Moodle` com a ferramenta de correção automática. Inicialmente, esta "solução"não passa os testes.

a) Corrigir a sub-rotina de forma a que passe o primeiro teste (usar a ferramenta para confirmar). Qual era a causa do erro?
Nota: pode usar o sistema DS-5 para perceber quais são os problemas. Valor absoluto

b) Corrigir a sub-rotina de forma a que passe o segundo teste. Qual era a causa do erro? CBZ no início

c) Corrigir a sub-rotina de forma a passar também o terceiro teste. Qual era a causa do erro? HI -> GE

Fim do enunciado