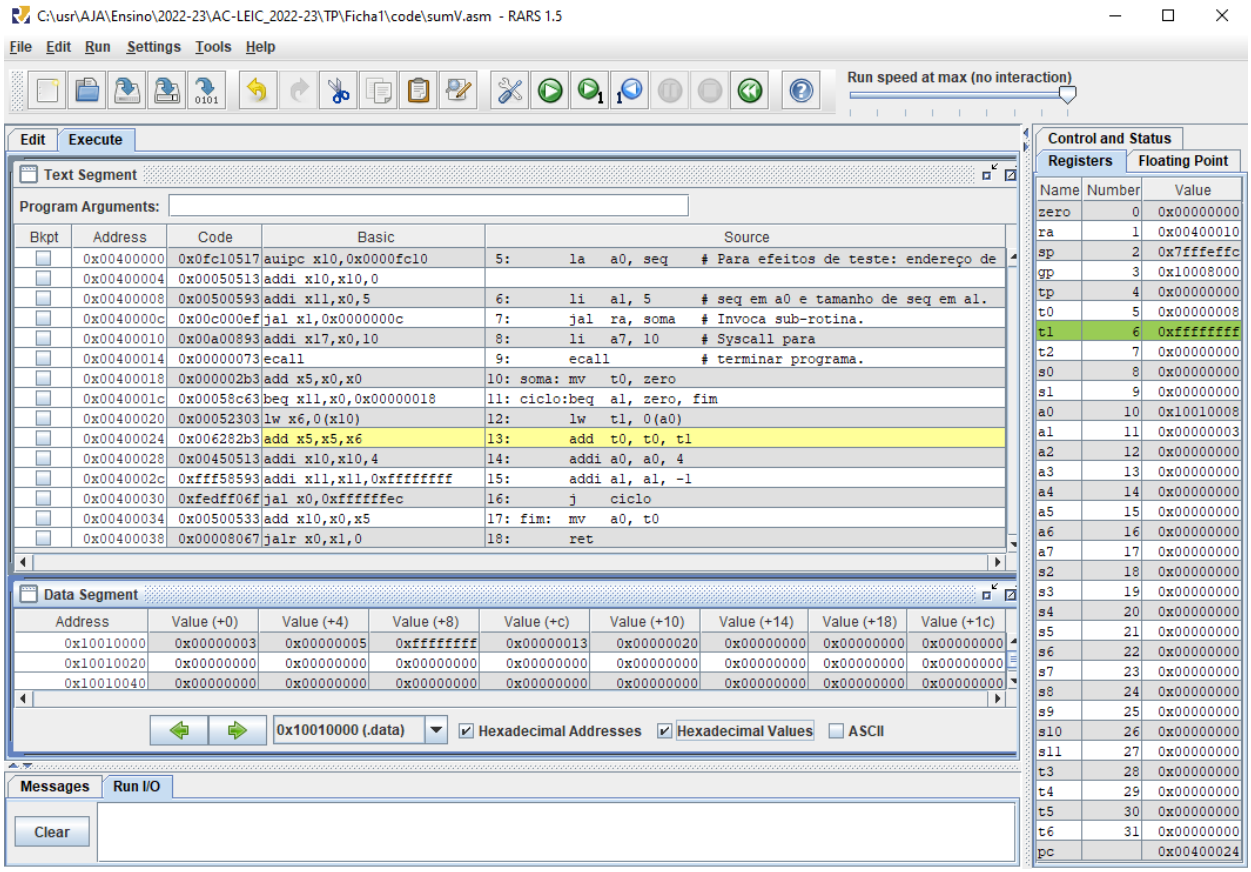
Ficha 1 – Sub-rotinas em RISC-V

1. A figura seguinte apresenta o estado de execução de uma sub-rotina no RARS. No teste apresentado, a sub-rotina processa uma sequência de cinco números inteiros (32 bits) armazenados em memória a partir do endereço `0x10010000`.

Obs.: O ficheiro ex1-sumV.asm, com o código desta sub-rotina e dados para teste, está disponível no Moodle.



a) Determine quantos bytes de memória são ocupados pela sequência de elementos.

b) Indique o valor decimal do quarto elemento da sequência.

c) Indique o endereço de memória do terceiro elemento da sequência.

d) Indique o endereço de memória da instrução `j ciclo`.

e) Tendo em consideração os valores dos registos, determine quantas vezes já foi executada a instrução `lw t1, 0(a0)`.

f) Determine o valor de `a0` após terminar a execução do código.

g) Descreva o que faz a sub-rotina.

2. Este exercício utiliza a ferramenta de correção automática FAST (Feup ASsembly correction Tool), disponível em `http://aoco.fe.up.pt` e em `http://arqcomp.fe.up.pt`.

Escrever uma sub-rotina que substitui por 0 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, do tipo `unsigned word`, pela ordem indicada:

1. valor `X`;
2. número de elementos da sequência;
3. endereço-base da sequência.

A sub-rotina seguinte pretende ser uma solução deste exercício mas, propositadamente, possui erros. Ao submeter esta solução à ferramenta de correção receberá mensagens de erro, devendo interpretá-las para corrigir tais erros.

```
.text
.global CheckABS

# a0 = valor "limite" X
# a1 = tamanho do vetor
# a2 = endereço do vetor

CheckABS:
        li   t0, 0          # Contador de ocorrências
iter:   lw   t1, 0(a2)      # Lê elemento do vetor
        bgtu t1, zero, pos
        neg  t1, t1         # Valor absoluto do elemento lido
pos:    bleu t1, a0, prox
        sw   zero, 0(a2)    # Como |elemento|>a0 então elemento torna-se 0
prox:   addi a2, a2, 4      # Atualiza endereço
        addi a1, a1, -1     # Atualiza iterações em falta
        bne  a1, zero, iter
        mv   a0, t0         # Coloca resultado em a0
        ret
```

Obs.: Este código está no ficheiro ex2-CheckABS.asm disponível no Moodle.

a) Um dos testes pré-definidos para esta sub-rotina considera o valor X=20 e a sequência de 8 elementos [7, -8, -23, 56, 20, -10, 0, 40]. Os resultados esperados são: número de elementos substituídos por 0 é 3 e a sequência modificada é [7, -8, 0, 0, 20, -10, 0, 0].
Execute o código. A primeira mensagem de erro indica que o resultado encontrado, 0 elementos modificados e sequência resultante [7, 0, 0, 0, 20, 0, 0, 0], não é o esperado. Portanto, o número de elementos modificados e a sequência resultante estão errados.
Identifique a causa destes erros, corrija o código e teste de novo até passar o primeiro teste.

b) O outro teste pré-definido na ferramenta considera o valor X=36 e uma sequência vazia (tamanho 0) e também falha. Identifique a causa do erro e altere o código de modo a passar o teste.

3. Implemente a sub-rotina `sumSquare` que recebe um inteiro n e, se positivo, retorna o valor da expressão seguinte e se não for positivo retorna 0.

$$n^2 + (n-1)^2 + (n-2)^2 + \cdots + 1^2$$

4. Considerar o seguinte programa, onde são invocadas duas sub-rotinas (`POLI` e `QUAD`).

```
.data
num: .word 7

.text
la   t0, num
lw   a0, 0(t0)
jal  ra, POLI
li   a7, 10
ecall       # termina
```

```
POLI: ...            # ***
      mv   s0, a0    # <1>
      jal  ra, QUAD
      mv   t0, a0
      li   a0, 3
      mul  s0, s0, a0
      add  a0, t0, s0
      addi a0, a0, 1
      ...            # ***
      ret            # <2>

QUAD: mul  a0, a0, a0
      ret
```

a) Analise o programa e descreva o que calcula a sub-rotina `POLI`.

b) Completar o código com as instruções que ache necessárias nos locais assinalados por ***.

c) Mostre o conteúdo da pilha do sistema imediatamente antes da execução das instruções assinaladas com <1> e <2>.

5. Nas alíneas seguintes pretende-se escrever e testar sub-rotinas que envolvem vetores com um ou mais elementos. Assumir que o endereço base do vetor e o número de elementos são passados à sub-rotina por esta ordem (caso existam mais argumentos devem ser passados a seguir aos anteriores).

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

b) Determinar o valor máximo de um vetor com dados do tipo `unsigned word`.

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

d) Determinar o número de valores de um vetor que pertencem ao intervalo $[a; b]$ assumindo dados do tipo `word`.

e) Contar quantos números pares existem num vetor de inteiros de 32 bits.

f) Inverter a ordem dos elementos de um vetor com inteiros de 32 bits.

6. Escrever e testar programas que usem uma sub-rotina para:

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

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

7. Pretende-se escrever sub-rotinas que executam tarefas envolvendo uma cadeia de carateres terminada por zero (`'\0'`). No caso de sub-rotinas que utilizem outras sub-rotinas, deverá aplicar (na *caller* e na *callee*) as regras de utilização e preservação de registos de modo a assegurar o correto funcionamento.

a) Determinar o comprimento de uma cadeia de carateres.

b) Determinar o número de ocorrências de um caráter numa cadeia de carateres.

c) Contar quantas palavras tem uma cadeia de carateres, assumindo que há um único espaço entre palavras consecutivas. Utilizar a sub-rotina da alínea b).

d) Determinar o número de vogais (minúsculas ou maiúsculas) de uma cadeia de carateres utilizando a sub-rotina da alínea b). Sugestão: coloque as vogais num *array* e, para cada uma, calcular o número de ocorrências na cadeia de carateres.

e) Determinar o número de letras maiúsculas de uma cadeia de carateres.

f) 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").
Determinar 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.

8. Considere uma sub-rotina que calcula a soma dos elementos de um vetor `v` com `n` elementos e cujo protótipo é "`int SOMA(int *v, int n);`".

Escreva as seguintes sub-rotinas considerando os respetivos protótipos em C. Para as testar deve utilizar a sub-rotina `SOMA` que o docente fornecerá.

a) A sub-rotina `MEDIA` calcula a média dos elementos de um vetor utilizando a sub-rotina `SOMA`.

```
int MEDIA(int *v, int n);
```

Obs.: Para realizar a divisão utilize a instrução `div rd,rs1,rs2` que calcula o valor de `rs1/rs2` (quociente da divisão inteira) e guarda-o em `rd`.

b) A sub-rotina `MAXMED` calcula a média dos elementos de dois vetores, v1 e v2, com n elementos, e retorna a média com maior valor.

```
int MAXMED(int *v1, int *v2; int n);
```

9. 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 sem sinal do tipo `word`.

b) Copiar um vetor com valores (com sinal) do tipo `halfword` para um vetor com valores do tipo `word`.

10. Escrever e testar as seguintes sub-rotinas em que é processado um argumento do tipo `word`.

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

b) Tendo em consideração a representação binária de um número inteiro, determine o número mínimo de bits em que pode ser representado.

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.

11. Considerar os vetores A, B e R, com n números inteiros de 32 bits, em que A e B são operandos e R é o vetor, ou escalar, resultante. Implementar e testar programas que implementem as seguintes operações vetoriais:

a) Adição de vetores ($R \leftarrow A + B$) – assumir que a soma de cada par de elementos, $A_i + B_i$, é representável com o mesmo número de bits dos operandos, ou seja, assumir que não ocorre *overflow*.

b) Adição de vetores ($R \leftarrow A+B$) – se ocorrer *overflow* ao somar dois valores usar como resultado o maior ou o menor valor representável (saturação), conforme essa soma seja, respetivamente, positiva ou negativa.

c) Multiplicação de um vetor por um inteiro (($R \leftarrow k \times A$)) – assumir a não ocorrência de *overflow*.

d) Produto interno ($R \leftarrow A \times B$) – assumir que não ocorre *overflow* durante o cálculo.

e) Produto interno ($R \leftarrow A \times B$) – caso ocorra *overflow*, na adição ou na multiplicação, 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

Algumas soluções:

1.

a) 20.

b) 19.

c) `0x10010008`.

d) `0x00400030`.

e) 3.

f) 58.

g) A sub-rotina calcula a soma dos elementos da sequência.

3.

```
sumSquare:
      addi sp, sp, -16  # Reserva espaço para 4 words na pilha
      sw   ra, 12(sp)   # Guarda endereço de retorno
      sw   s0, 8(sp)    # Guarda registos
      sw   s1, 4(sp)    # saved (s0 e s1)
      add  s0, a0, x0   # Copia valor de n para s0
      add  s1, x0, x0   # Inicializa s1 (irá acumular a soma)
loop: bge  x0, s0, end  # Salta if s0 não é positivo
      add  a0, s0, x0   # Copia s0 para a0 para invocar 'square'
      jal  ra, square   # Chama a sub-rotina 'square'
      add  s1, s1, a0   # Atualiza s1 com o valor retornado em a0
      addi s0, s0, -1   # Decrementa s0 de 1
      jal  x0, loop     # Salta para o início do ciclo
end:  add  a0, s1, x0   # Copia para a0 a soma acumulada em s1
      lw   ra, 12(sp)   # Recupera ra
      lw   s0, 8(sp)    # Recupera s0
      lw   s1, 4(sp)    # Recupera s1
      addi sp, sp, 16   # Liberta o espaço reservado na pilha
      jr   ra           # Retorna à sub-rotina que a invocou

square:
      mul  a0, a0, a0
      ret
```

4.

a) A sub-rotina POLI calcula o valor de $x^2 + 3x + 1$, sendo x o argumento que lhe é passado.

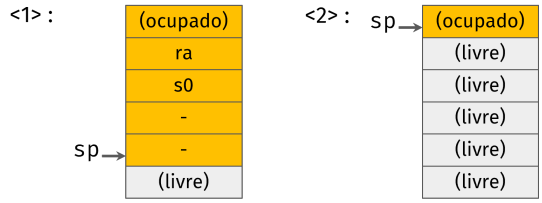
b) No início:

```
POLI: addi  sp, sp, -16
      sw    ra, 12(sp)
      sw    s0, 8(sp)
```

Antes de `ret`:

```
        lw    ra, 12(sp)
        lw    s0, 8(sp)
        addi  sp, sp, 16
```

c)



6.

a)
```
nbits1:
        mv    t0, zero
ciclo:  beq   a0, zero, fim
        andi  t1, a0, 1
        add   t0, t0, t1
        srli  a0, a0, 1
        j     ciclo
fim:    mv    a0, t0
        ret
```