

CESAR

exercícios e exemplos

Programa exemplo

- somar (totalizar) n posições consecutivas de memória, a partir do endereço inicial e .
 - sem consistência sobre os valores de n e e .
- em alto nível, o programa seria:

```
total := 0
ponteiro := e
contador := n
laço: if contador = 0, termina
      total := total + mem(ponteiro)
      ponteiro := ponteiro + 1
      contador := contador - 1
      goto laço
```

Endereço	Instrução	
0	LDA 132	; inicializa (zera) o total
2	STA 130	
4	LDA 129	; inicializa ponteiro
6	STA 17	
8	LDA 128	; inicializa contador
10	STA 131	
12	JZ 34	; testa se contador é zero
14	LDA 130	; carrega total no acumulador
16	ADD 17	; soma com posição de memória
18	STA 130	; atualiza total
20	LDA 17	; incrementa ponteiro
22	ADD 134	
24	STA 17	
26	LDA 131	; decrementa contador
28	ADD 133	
30	STA 131	
32	JMP 12	; retorna ao início do laço
34	HLT	
128	n	número de posições
129	e	endereço inicial
130	tot	total
131	cont	contador
132	0	constante zero
133	255	constante -1
134	1	constante 1

Neander

Implementação Ramses

Endereço	Instrução	
0	LDR A #0	; inicializa (zera) o total
2	LDR X 129	; inicializa ponteiro
4	LDR B 128	; inicializa contador
6	JZ 16	; testa se contador é zero
8	ADD A 0,X	; soma com posição de memória
10	ADD X #1	; incrementa ponteiro
12	SUB B #1	; decrementa contador
14	JMP 6	
16	STR A 130	; atualiza total
18	HLT	; retorna ao início do laço
128	n	número de posições
129	e	endereço inicial
130	total	total

Implementação Cesar

tradução literal de cada instrução

0	MOV #0, R0	; inicializa (zera) o total
4	MOV 1026, R2	; inicializa ponteiro
8	MOV 1024, R1	; inicializa contador
12	BEQ +14	; testa se contador é zero
14	ADD (R2), R0	; soma com posição de memória
16	ADD #2, R2	; incrementa ponteiro
20	SUB #1, R1	; decrementa contador
24	JMP 12	; retorna ao início do laço
28	MOV R0, 1028	; atualiza total
32	HLT	
1024	n	número de posições
1026	e	endereço inicial
1028	total	

Implementação Cesar

- Usando instruções específicas do Cesar

Endereço	Instrução	
0	CLR R0	; inicializa (zera) o total
2	MOV 1026, R2	; inicializa ponteiro
6	MOV 1024, R1	; inicializa contador
10	BEQ 10	; testa se contador é zero
12	ADD (R2), R0	; soma com posição de memória
14	ADD #2, R2	; incrementa ponteiro
18	DEC R1	; decrementa contador
20	BR -12	; retorna ao início do laço
22	MOV R0, 1028	; atualiza total
26	HLT	
1024	n	número de posições
1026	e	endereço inicial
1028	total	

Implementação Cesar

• Usando instruções específicas do Cesar

Endereço	Instrução	
0	CLR R0	; inicializa (zera) o total
2	MOV 1026, R2	; inicializa ponteiro
6	MOV 1024, R1	; inicializa contador
10	BEQ 10	; testa se contador é zero
12	ADD (R2), R0	; soma com posição de memória
14	ADD #2, R2	; incrementa ponteiro
18	DEC R1	; decrementa contador
20	BNE -10	; retorna ao início do laço
22	MOV R0, 1028	; atualiza total
26	HLT	
1024	n	número de posições
1026	e	endereço inicial
1028	total	

Implementação Cesar

• Usando endereçamento pós incrementado

Endereço	Instrução	
0	CLR R0	; inicializa (zera) o total
2	MOV 1026, R2	; inicializa ponteiro
6	MOV 1024, R1	; inicializa contador
10	BEQ +6	; testa se contador é zero
12	ADD (R2)+, R0	; soma com posição de memória ; e incrementa ponteiro
14	DEC R1	; decrementa contador
16	BNE -6	; retorna ao início do laço
18	MOV R0, 1028	; atualiza total
22	HLT	
1024	n	número de posições
1026	e	endereço inicial
1028	total	

Implementação Cesar

• Usando instrução SOB

Endereço	Instrução	
0	CLR R0	; inicializa (zera) o total
2	MOV 1026, R2	; inicializa ponteiro
6	MOV 1024, R1	; inicializa contador
10	BEQ +4	; testa se contador é zero
12	ADD (R2)+, R0	; soma com posição de memória ; e incrementa ponteiro
14	SOB R1, 4	; decrementa contador ; retorna ao início do laço
16	MOV R0, 1028	; atualiza total
20	HLT	
1024	n	número de posições
1026	e	endereço inicial
1028	total	

Escrita no Daedalus

	CLR R0	; inicializa (zera) o total
	MOV e, R2	; inicializa ponteiro
	MOV n, R1	; inicializa contador
	BEQ fim	; testa se contador é zero
somatorio:	ADD (R2)+, R0	; soma com posição de memória ; e incrementa ponteiro
	SOB R1, somatorio	; decrementa contador e ; retorna ao início do laço
fim:	MOV R0, total	; atualiza total
	HLT	
	ORG 1024	
n:	DW 0	; número de posições
e:	DW 0	; endereço inicial
total:	DW 0	; totalizacao

Soma de variáveis de 32 bits

1024	Bits mais significativos da primeira variável
1026	Bits menos significativos da primeira variável
1028	Bits mais significativos da segunda variável
1030	Bits menos significativos da segunda variável
1032	Bits mais significativos do resultado
1034	Bits menos significativos do resultado

Soma de variáveis de 32 bits

Uma solução possível

```

MOV 1026, R0 ; Bits menos significativos da primeira variável
ADD 1030, R0 ; Soma com bits menos significativos da segunda variável
MOV R0, 1034 ; Salva resultado da soma (nos bits menos significativos )
MOV #0, R0 ; Zera o registrador R0 (prepara para receber o carry)
ADC R0 ; Soma o carry da soma anterior
ADD 1024, R0 ; Soma com bits mais significativos da primeira variável
ADD 1028, R0 ; Soma com bits mais significativos da segunda variável
MOV R0, 1032 ; Salva o resultado (bits mais significativos)
HLT

```

Movimento de blocos de n posições

- faça um programa para mover (sem zerar a origem) um número qualquer de posições consecutivas na memória
 - o número de posições é determinado pelo conteúdo da posição 1024 de memória
 - a posição inicial do bloco de memória a ser movido é dada pelo conteúdo da posição 1026 de memória
 - o endereço inicial do bloco de destino é dado pela posição 1028
 - posição 1024: número de posições
 - posição 1026: posição inicial da origem
 - posição 1028: posição inicial do destino

Uma solução possível programa independente de posição

```

MOV 1024, R0 ; Tamanho do bloco (em palavras)
MOV 1026, R1 ; Endereço inicial da origem
MOV 1028, R2 ; Endereço inicial do destino
CMP R2, R1 ; Compara endereço de destino com o de origem
BGT 5 ; Desvia de end.destino > end.origem
MOV (R1)+, (R2)+ ; Move uma palavra no sentido crescente
SOB R0, 4 ; Laço para mover toda a área
HLT ; Fim do programa
ASL R0 ; Multiplica tamanho por dois (obtem tam. em bytes)
ADD R0, R1 ; Endereço final da origem (+ 2 bytes)
ADD R0, R2 ; Endereço final do destino (+ 2 bytes)
MOV 1024, R0 ; Restaura tamanho para palavras
MOV -(R1), -(R2) ; Move uma palavra no sentido decrescente
SOB R0, 4 ; Laço para mover toda a área
HLT
    
```

Pesquisa em vetores

- faça um programa para determinar o maior valor armazenado em um vetor (array)
 - o tamanho do vetor é determinado pelo conteúdo da posição 1024 de memória e a posição inicial do vetor é dada pelo conteúdo da posição 1026. O maior valor encontrado deve ser colocado na posição 1028, e a posição relativa desse valor no vetor (1º, 2º, ..., n-ésimo) na posição 1030.
 - posição 1024: número de posições (tamanho do vetor)
 - posição 1026: posição inicial do vetor
 - posição 1028: resultado: maior valor encontrado
 - posição 1030: resultado: posição relativa do maior valor

Pesquisa em vetores

mapa dos registradores

registrador	função	comentário
R0	contador	inicializado com tamanho do vetor
R1	endereço do elemento do vetor	inicializado com endereço do vetor
R2	maior elemento atual	
R3	posição do maior elemento	
R4	índice do maior elemento no final	

Uma solução possível programa independente de posição

```

MOV 1024, R0 ; Tamanho do vetor (em palavras)
MOV 1026, R1 ; Endereço inicial do vetor
MOV (R1)+, R2 ; Inicializa o primeiro elemento como sendo o maior
MOV R0, R3 ; Inicializa R3 com compl. índice ("tamanho") do maior elemento
DEC R0 ; Inicializa contador (tamanho - 1)
CMP (R1), R2 ; Compara um elemento com o maior atual
BLE 4 ; Desvia se for menor ou igual
MOV (R1), R2 ; Se for maior, atualiza R2
MOV R0, R3 ; Salva índice do novo maior valor ("contador atual")
ADD #2, R1 ; Em qualquer caso, incrementa ponteiro
SOB R0, 14 ; Controle do laço
MOV R2, 1028 ; Fornece maior valor encontrado
MOV 1024, R4 ; Calcula índice do maior valor
SUB R3, R4 ; índice = tamanho - contador + 1
INC R4 ; índice = mem(1024) - R3 + 1
MOV R4, 1030 ; Fornece o índice do maior valor
HLT
    
```

Alteração de bits

- escreva um programa que zere (clear) ou ligue (set) um bit qualquer de uma palavra qualquer da memória, conforme indicado por um parâmetro na memória.

posição 1024: endereço da palavra a ser alterada
 posição 1026: posição do bit a ser alterado (0 é o lsb)
 posição 1028: conteúdo = 0, para zerar
 conteúdo = 1, para ligar

Uma solução possível

```
MOV 1026, R1 ; Obtém índice do bit a ser alterado
MOV 1024, R2 ; Obtém endereço da palavra a ser alterada
JSR R7, 100 ; Chama a subrotina de geração da máscara
TST 1028 ; Testa se o bit deve ser ligado ou desligado
BEQ 6 ● ; Deve ser desligado
OR R0, (R2) ; Deve ser ligado, usar OR
BR 4 ● ; Vai para o fim do programa
NOT R0 ; Desligar bit: inverte a máscara
● AND R0, (R2) ; Desliga o bit usando AND
● HLT

No endereço 100 (programa dependente de posição de carga na memória):
MOV #1, R0 ; R0 contém a máscara (inicializada com 0000000000000001)
TST R1 ; R1 contém o índice do bit a ser isolado (0 é o bit menos sign.)
BEQ 4 ● ; Se o índice é zero, a máscara está pronta
● ASL R0 ; Desloca o bit da máscara para esquerda
SOB R1, 4 ● ; Decrementa o índice e desloca a máscara até o índice ser zero
● RTS R7 ; Retorna ao programa principal (máscara em R0)
```