; E-fólio B - alocação de memória dinâmica

ORIG 4000h

; definir o HEAP a meio (o programa está no início)

HEAP TAB 8000h

; variáveis do programa de teste

ORIG 1000h

Passos WORD 32

Falhas WORD 0

ChamadasMALLOC WORD 0

ChamadasFREE WORD 0

; ponteiros retornados

Ponteiros TAB 4

; instruções de alocação e libertação de memória, a executar

Tamanho STR 16243, 8214, 14874, 0, 0, 0, 3936, 8907, 12908, 0, 0, 14443, 0, 0, 1248, 15723, 0, 4587, 0, 339, 11275, 11809, 0, 0, 0, 0, 3710, 14428, 0, 0, 0, 0

Variavel STR 2, 1, 3, 2, 3, 1, 2, 3, 0, 0, 2, 0, 0, 3, 1, 0, 0, 3, 3, 2, 3, 0, 0, 1, 2, 3, 1, 0, 0, 1, 2, 3

ORIG 0000h

; colocar o STACK no local habitual das AFs (no final)

MOV R1, fd1fh

MOV SP, R1

CALL INITHEAP ; inicializar o HEAP

; Programa de teste:

; R1 - registo para passagem e retorno de argumentos do MALLOC/FREE

; R2 - registo para guardar o passo atual

; R3 - apontador a atribuir/ler

MOV R2, R0 ; inicializar algoritmo de teste

DEC R2

TESTEpasso: INC R2 ; próximo passo

CMP R2, M[Passos]

BR.Z TESTEfinal ; fim do teste

MOV R1, M[R2+Tamanho] ; ler tamanho

CMP R1, R0

BR.Z TESTEfree ; pretende-se libertar e não alocar

CALL MALLOC ; aloca os elementos

MOV R3, M[R2+Variavel]

MOV M[R3+Ponteiros], R1 ; guardar o valor do apontador retornado

CMP R1, R0

BR.NZ TESTEpasso ; operação bem sucedida

INC M[Falhas] ; incrementa o número de falhas, já que R1=0

BR TESTEpasso

TESTEfree: MOV R3, M[R2+Variavel]

MOV R1, M[R3+Ponteiros] ; libertar o apontador guardado previamente

CALL FREE

BR TESTEpasso

TESTEfinal: MOV R2, 8 ; Copiar os primeiros elementos do HEAP para primeira posições em Tamanho

TESTEpassoB: DEC R2

MOV R4, M[R2+HEAP]

MOV M[R2+Tamanho], R4

BR.NZ TESTEpassoB

Fim: JMP Fim

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Operação: MALLOC

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Argumentos: R1 – número de posições de memória

; Resultado: R1 – endereço do início do bloco (0/falha)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MALLOC: INC M[ChamadasMALLOC]

; R1 argumento, com o número de posições N

; R2 bloco atual

; R3 tamanho do bloco atual

; R4 auxiliar

PUSH R2

PUSH R3

PUSH R4

MOV R2, HEAP ; primeiro bloco

MALLOCteste: MOV R3, M[R2] ; ler a posição de controlo

CMP R3, R0

BR.Z MALLOCfalha ; último bloco, não há espaço

BR.P MALLOCprox ; bloco ocupado, ir para o próximo

; bloco livre

NEG R3

CMP R3, R1

BR.N MALLOCprox ; sem espaço, já que tamanho<R1

; bloco suficiente, ver a diferença

MOV R4, R3

SUB R4, R1

CMP R4, 1 ; se a diferença for menor ou igual a 1, não criar um novo bloco

BR.P MALLOCnovo

NEG M[R2] ; bloco alocado

BR MALLOCfim

MALLOCprox: ADD R2, R3 ; adicionar ao bloco atual o tamanho do bloco +1

INC R2

BR MALLOCteste

MALLOCfalha: MOV R2, R0 ; operação falhada

DEC R2 ; força o retorno de R1=0

BR MALLOCfim

MALLOCnovo: MOV M[R2], R1 ; bloco alocado, criar o bloco vazio

MOV R3, R2

ADD R3, R1

INC R3

DEC R4 ; R4 tem a diferença de tamanhos, mas perde-se uma posição para controlo

NEG R4 ; basta negar para criar um bloco livre

MOV M[R3], R4 ; bloco vazio criado

MALLOCfim: MOV R1, R2

INC R1

POP R4

POP R3

POP R2

RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Operação: FREE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Argumentos: R1 – endereço do início do bloco

; Resultado: nenhum, apenas liberta o bloco

; previamente devolvido por MALLOC

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FREE: INC M[ChamadasFREE]

; R2 tamanho do bloco

; R3 bloco seguinte

; R4 tamanho do bloco seguinte/anterior

PUSH R2

PUSH R3

PUSH R4

; não fazer nada se R1=0

CMP R1, R0

JMP.Z FREEfim

DEC R1 ; R1 = posição de controlo

MOV R2, M[R1] ; R2 = número de posições de memória

NEG M[R1] ; bloco liberto

JMP.P FREEerroA ; libertar o bloco duas vezes

; procurar juntar à direita

MOV R3, R1

ADD R3, R2

INC R3 ; R3 está na posição de controlo do próximo bloco

MOV R4, M[R3] ; R4 é o tamanho do bloco seguinte

CMP R4, R0

BR.NN FREEesq ; caso o tamanho não seja negativo, o bloco não está livre

; juntar com o bloco da direita

NEG R4

ADD R2, R4 ; R2 fica com o tamanho de ambos os blocos

INC R2 ; remove-se um bloco, ganha-se uma posição de controlo

MOV M[R1], R2 ; bloco adicionado

NEG M[R1]

FREEesq: MOV R3, HEAP ; procurar deste do início, pelo que R3 começa do primeiro bloco

CMP R3, R1 ; se R1 é a posição inicial, nada a juntar à esquerda

BR.Z FREEfim

FREEprox: CMP R3, R1

BR.Z FREEc ; R3 chegou até R1, o bloco a libertar

; avançar para o próximo bloco

MOV R4, M[R3]

CMP R4, R0

BR.Z FREEerror

BR.NN FREEa

NEG R4

FREEa: ADD R3, R4 ; R4 é o tamanho do bloco atual, seja ocupado ou livre

INC R3

BR FREEprox

FREEerror: NOP

BR FREEfim

FREEerroA: NEG M[R1] ; repor o bloco para manter-se livre e não fazer nada

BR FREEfim

FREEc: DEC R3 ; R3 é igual a R1, recuperar a anterior posição

SUB R3, R4

MOV R4, M[R3] ; R4 com o novo tamanho do bloco anterior

CMP R4, R0

BR.NN FREEfim ; está alocado, nada a fazer

NEG R4

ADD R2, R4

INC R2

MOV M[R3], R2 ; bloco fundido

NEG M[R3]

FREEfim: POP R4

POP R3

POP R2

RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Operação: INITHEAP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Argumentos: nenhum

; Resultado: nenhum, inicializa o HEAP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

INITHEAP: PUSH R2 ; variável temporária

MOV R2, 8000h

SUB R2, 2 ; memória utilizável

MOV M[HEAP], R2 ; bloco alocado com toda a memória

NEG M[HEAP]

INC R2

MOV M[R2+HEAP], R0 ; último bloco terminador

POP R2

RET