E-fólio B | Instruções para a realização do E-fólio

21010 – Arquitectura de Computadores

Enunciado

Pretende-se implementar no P3 a alocação de memória dinâmica, através das rotinas MALLOC e FREE. Ambas as rotinas utilizam R1 para entrada e saída de parâmetros. A rotina MALLOC recebe em R1 o número de elementos necessários, e é retornado o apontador para a primeira posição de memória com o bloco de elementos alocado. A rotina FREE recebe o apontador previamente retornado pelo MALLOC e liberta o bloco de memória. No caso de não existir espaço disponível, a função MALLOC retorna 0. Existe ainda uma função INITHEAP, que é chamada no início do programa, para inicializar a gestão de memória.

Vamos implementar algumas soluções nas quatro alíneas. O mesmo programa de teste irá funcionar para todas as alíneas. Este programa tem declarado o bloco de memória que será gerido, de nome HEAP, a iniciar-se na posição de memória 4000h, tendo 8000h posições. Contém também algumas variáveis do programa na zona de memória 1000h, e chama as funções INITHEAP, MALLOC e FREE simulando a execução de um programa.

Apenas o código das rotinas deve ser editado.

a) **[1]** Vamos na alínea a) e b), considerar que metade da memória da HEAP é utilizada para controlo, as primeiras 4000h posições. Cada posição de memória de controlo, tem a correspondente associada com 4000h posições à frente. Uma posição de controlo indica se a posição que referencia, está em utilização (>0) ou não (=0), e no caso de estar em utilização, tem o número de posições do bloco a que pertence.

Exemplo (HEAP com 8 posições de memória):

0	1	2	3	4	5	6	7
2	2	1	0	253	6474	2532	277

Temos no exemplo a verde, a zona de memória de controlo (a primeira metade), a azul é a memória que será utilizada pelos programas. Pode-se ver que na situação atual, as três primeiras posições de memória estão alocadas, já que a zona de controlo é distinta de zero, e a última posição está livre, já que a zona de controlo é zero. Temos ainda a informação que as duas primeiras posições foram alocadas com um bloco de 2 elementos, e a terceira posição foi alocada com um bloco de 1 posição de memória.

Assim, deve em:

1. INITHEAP: iniciar as primeiras 4000h posições de memória a 0

Exemplo de 8 posições de memória, após chamada de INITHEAP:

0	1	2	3	4	5	6	7
0	0	0	0	253	6474	2532	277

A zona a verde é toda zerada, o resto da memória fica como estava.

b) [1] Nesta alínea deve continuar a implementação iniciada na alínea a):

2. MALLOC: ver o tamanho N solicitado (em R1), e procurar pelo primeiro conjunto de N posições na zona de controlo, que estejam todas a zero. Retornar o endereço correspondente da memória em utilização (4000h posições acima). Marcar todas as posições utilizadas com o valor N.

Exemplo de 8 posições de memória, após chamada MALLOC com R1=3:

0	1	2	3	4	5	6	7
3	3	3	0	253	6474	2532	277

A zona a verde é atualizada, e em R1 fica com o valor 4, que é o início do bloco alocado de três elementos. Notar que neste caso a primeira posição na zona de controlo é 0, mas foi retornado 4, que é a primeira posição na zona de utilização.

3. FREE: recebe a posição de memória, e após ver qual o tamanho do bloco, zera todas as posições referentes ao bloco de memória alocado.

Exemplo de 8 posições de memória, após chamada FREE com R1=4:

0	1	2	3	4	5	6	7
0	0	0	0	11	22	33	277

A zona a verde é atualizada pelo FREE. Na zona azul tinham sido utilizadas e alteradas pelo programa as posições de memória 4 a 6, que foram alocadas. Após a chamada do FREE, a zona de controlo desse bloco de memória ficou a zero.

c) [1] Vamos agora procurar melhorar a solução atual, já que a mesma necessita de 50% da memória para controlo. Vamos desta vez alocar um bloco de tamanho N+1 sempre que for solicitado um bloco de tamanho N. A primeira posição do bloco, é uma posição e controlo, sendo devolvido ao programa o endereço da segunda posição. Na primeira posição de memória, encontra-se o tamanho do próprio bloco, permitindo assim ao FREE libertar o bloco. Os blocos de memória libertados são considerados com tamanho negativo. Assim, de início existe um bloco com toda a memória, mas de tamanho negativo para indicar que toda a memória está livre. Existe ainda um bloco final, de tamanho nulo, de modo a facilitar o teste de falta de memória. O MALLOC deve processar cada bloco à vez, em vez de processar cada posição de memória.

Exemplo (HEAP com 8 posições de memória):

0	1	2	3	4	5	6	7
23	252	461	102	289	3464	4722	387

Assim, deve em:

1. INITHEAP: iniciar um bloco de memória livre com toda a memória menos duas posições, portanto 8000h-2 posições utilizáveis, e um último bloco com 0 posições.

Exemplo (HEAP com 8 posições de memória):

0	1	2	3	4	5	6	7
-6	252	461	102	289	3464	4722	0

O primeiro bloco fica com toda a memória, exceto naturalmente as posições de controlo. O sinal negativo é uma indicação de que o bloco está livre.

2. MALLOC: processar todos os blocos por ordem, a começar pelo **primeiro**, vendo todas as posições de controlo. Parar assim que for localizado um bloco com espaço suficiente. Se o tamanho for igual (ou ser apenas uma unidade maior), marcar o bloco como alocado, caso contrário criar um bloco com o resto do bloco que se mantém livre, alocando no início do bloco.

Exemplo de 8 posições de memória, após chamada MALLOC com R1=3:

0	1	2	3	4	5	6	7
3	252	461	102	-2	3464	4722	0

É criado num bloco de 3 posições de memória, R1 retorna com o valor 1. O bloco ocupa 4 posições, já que uma é utilizada para controlo. Resta um bloco livre com duas posições. Vamos ver o resultado após chamar MALLOC com R1=2:

0	1	2	3	4	5	6	7
3	555	33	88	2	3464	4722	0

É retornado R1=5, e a operação foi bem-sucedida. Pode-se ver que neste exemplo, o programa já fez uso do primeiro bloco que alocou.

3. FREE: aceder à primeira posição e colocar a posição a negativo.

Exemplo de 8 posições de memória, após chamada FREE com R1=1:

0	1	2	3	4	5	6	7
-3	555	33	88	2	3464	4722	0

O primeiro bloco é libertado, ficando disponível para ser alocado. Ao libertar-se também o segundo bloco, com R1=5, fica:

0	1	2	3	4	5	6	7
-3	555	33	88	-2	3464	4722	0

A memória fica novamente toda livre, com dois blocos livres.

Este método deve estar preparado para as seguintes utilizações incoerentes:

- Chamar o método com R1=0, portanto um argumento não válido. Não deve fazer nada.
- O endereço fornecido tem na posição de controlo um valor não positivo, resultante por exemplo de chamar a função FREE duas vezes para o mesmo apontador. Não deve também fazer nada nessa situação.

d) [1] Melhore o algoritmo do método FREE da alínea anterior, de modo a juntar o bloco liberto com o bloco da direita e/ou da esquerda, caso esses blocos estejam também livres. Note que para localizar o bloco da esquerda, tem de iterar desde o início.

Na execução anterior da alínea c), com R1=5, ficaria neste caso:

0	1	2	3	4	5	6	7
-6	555	33	88	2	3464	4722	0

A memória fica novamente toda livre, com um único bloco livre.

Programa de teste:

```
; 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]
                RET


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Operação: FREE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Argumentos: R1 - endereço do inicio do bloco
; Resultado: nenhum, apenas liberta o bloco
;            previamente devolvido por MALLOC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FREE:           INC M[ChamadasFREE]
                RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Operação: INITHEAP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Argumentos: nenhum
; Resultado: nenhum, inicializa o HEAP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INITHEAP:       RET
```

Resultado de execução do programa de teste sem qualquer alteração:

```
Clock: 4442
Instructions: 442
1000 : 0020 0000 000f 0011 385c 0e7e 0153 2c0b ........
1008 : 0000 0000 0000 0000 0000 0000 0000 0000 ........
```

BOM TRABALHO!

Avaliação

Cotação:
A cotação encontra-se junto de cada uma das alíneas, entre [].

Critérios de Correcção:
Funcionalidade: 50%
Simplicidade e Modularidade: 10%
Eficiência (serão contabilizados o número de instruções e ciclos de relógio): 10%
Apresentação do código (indentação e comentários): 20%
Relatório (Legibilidade e Justificação dos Resultados e das Opções): 10%

Descontos:
Trabalhos entregues que não estejam em conformidade com as regras de entrega do e-fólio B: até 10%
Código sem comentários, ou apenas com comentários a reflectir o significado da instrução (exemplo MOV R1,R2 ;mover o conteúdo de R2 para R1) : até 50%
Detecção de fraude (total ou parcial): 100%
Trabalhos entregues após a data limite (máximo 24h) : 10%

Regras para entrega do e-fólio B:

Forma de entrega:

Deverá ser entregue um relatório em formato pdf ou Word até 5 páginas A4, com todos os cálculos e todas as opções tomadas na construção dos programas. Em anexo deve colocar todo o código (apenas o código das rotinas) e resultados obtidos (número de instruções, ciclos de relógio e conteúdo da memória de 1000h a 1000fh). O código deve estar num formato que permita a seleção de modo a ser copiado e colado para o simulador do P3.

Não são aceites entregas fora da plataforma Moodle.