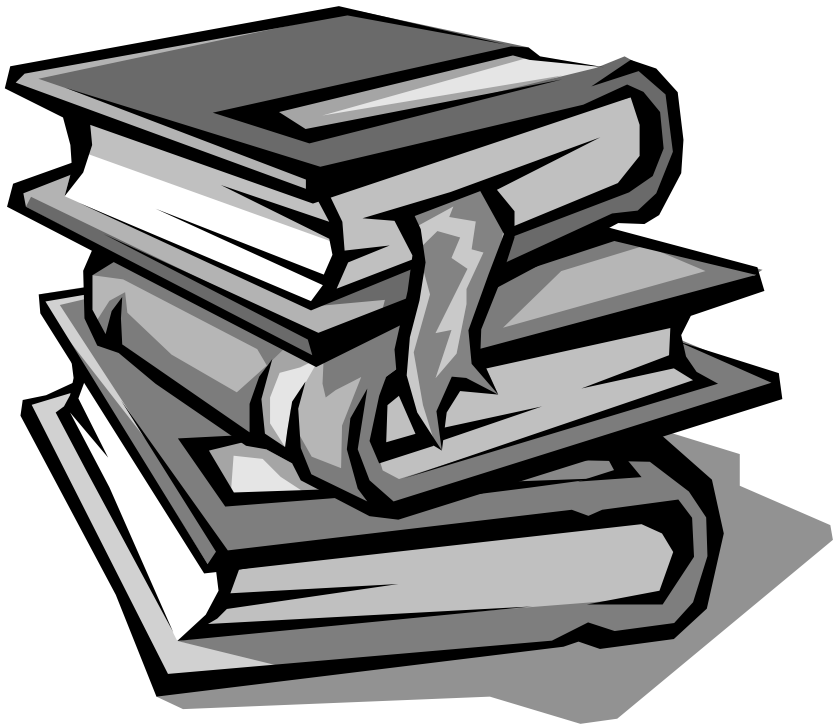
INSTITUTO POLITÉCNICO DE BRAGANÇA
ESCOLA SUPERIOR DE TECNOLOGIA E DE GESTÃO

Sistemas Digitais II



Filipe Moreira

Exercícios sobre Assembly Z80

Exercícios gerais de dificuldade baixa e média

1. Escrever um programa que coloque 1h no registo D, 56h no registo C, 24h no registo B e nos endereços 4000h...400Fh coloque os valores 0h...Fh, respectivamente.

2. Escrever um programa que efectue a subtracção dos registos B e C, colocando o resultado na posição de memória 4000h. Suponha que B=203 e C=120.

3. Escrever um programa que adicione um número de 16 bits guardado nos endereços 4000h (*low*) e 4001h (*high*) com o número existente no par de registos DE. O resultado final é colocado em HL.

4. Escrever um programa que mova 17 bytes colocados a partir do endereço 4000h para os endereços a partir de:
   a) 4040h
   b) 4003h

5. Escrever um programa que faça a soma de dois números com três bytes cada um. O primeiro está contido nas posições 4050h, 4051h e 4052h e o segundo em 4053h, 4054h e 4055h. O resultado é colocado a partir de 4056h.

6. Escrever um programa que faça a subtracção de dois números com três bytes cada um. O primeiro está contido nas posições 4050h, 4051h e 4052h e o segundo em 4053h, 4054h e 4055h. O resultado é colocado a partir de 4056h.

7. Efectuar a soma de 16 bytes colocados a partir da posição de memória 4000h. O resultado é guardado no par DE.

8. Deslocar os bytes colocados a partir do endereço 4000h (inclusive) para os endereços a partir de 4Aooh, limpando os bits 0, 1, 2 e 3 e deixando os restantes com o valor inicial. A deslocação termina quando se encontra o valor 0FFh.

9. Escrever a rotina MAXMIN que determina o máximo e o mínimo entre o conteúdo de duas posições de memória apontadas por HL e DE. O máximo deve ser colocado em B e o mínimo em C. Considerar os conteúdos referidos como valores positivos (pertencentes ao intervalo [0,255]).

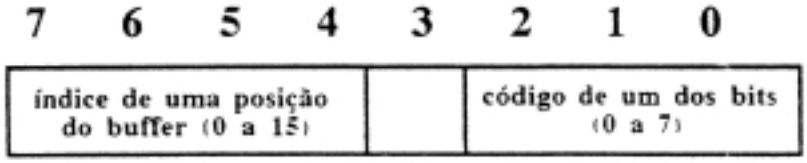
10. Utilizar a rotina anterior, MAXMIN, na escrita de um programa que determina o máximo do conteúdo de 12 posições de memória situadas a partir de 4000h. Colocar o valor máximo em 400Fh.

11. Escrever uma variante do programa anterior em que se determina o máximo e o mínimo só para os conteúdos com um número par de 1's. O máximo é colocado em 400Fh e o mínimo em 4010h.

12. Escrever uma rotina variante de MAXMIN (MM2) que actualiza o máximo e o mínimo em B (Max.) e C (Min.) com o conteúdo da posição apontada por HL.

13. Escrever um programa que determina o máximo e o mínimo de um conjunto de posições de memória consecutivas com início na posição apontada por HL e de comprimento igual ao conteúdo de D.
Nota: utilizar a rotina anterior MM2.

14. Desconhece-se o conteúdo de um *buffer* em memória constituído por 16 posições consecutivas, a primeira das quais apontada por HL.
A rotina ACTUAL (actualiza) começa por limpar o *buffer* referido e, seguidamente, actualiza o respectivo conteúdo de acordo com um código situado no registo A.



Os bits de 7 a 4 identificam uma posição do *buffer* através de um índice (0 a 15).
Os bits de 2 a 0, através de um código (0 a 7), identificam um dos bits da posição referida pelos bits 7 a 4.

a) Escrever em Assembly, devidamente comentada, a rotina ACTUAL, sabendo que o bit encontrado, através do código situado no registo A, é colocado a 1.
b) Alterar a rotina ACTUAL para que, no bit encontrado, fosse colocado o valor do bit 3 do registo A.

Exemplo: supor que HL aponta para a posição 4000h e o conteúdo de A é 35h. 3 de 35h identificará a posição 4003h e 5, do mesmo valor 35h, identificará o bit 5. Chamando a rotina ACTUAL, as 16 posições a partir de 4000h ficarão com o valor 00h, excepto a4003h que conterá o valor 20h no caso da alínea a), ou 00h no caso da alínea b).

15. A primeira posição de 2 *buffers* em memória (BF1 e BF2) é apontada por AP1 (para BF1) e por AP2 (para BF2); AP1 e AP2 são apontadores de 16 bits, implementados em memória. Estes *buffers* memorizam números positivos (de 0 a 255) já ordenados do menor para o maior. Os comprimentos de BF1 e BF2 estão colocados nas posições de memória C1 e C2, respectivamente.
Apresentar um fluxograma e escrever em Assembly (Z80), devidamente comentada, a rotina AGLUTI (aglutina) que, a partir daqueles 2 *buffers* cria um terceiro (BF), apontado por AP (também um apontador em memória) e cujo comprimento, na posição de memória COM, será a soma dos comprimentos de BF1 e BF2. O conteúdo de BF apresentar-se-á também ordenado do menor para o maior valor.

16. Escrever a rotina P.BYTE que procura todas as ocorrências de um byte (cujo valor se encontra no registo A), num *buffer* apontado por HL e de comprimento em BC. Os

endereços são guardados num segundo *buffer*, apontado inicialmente por DE. Caso não seja encontrado qualquer byte, este *buffer* terá a sua primeira posição (2 bytes) igual a 0h.

17. Num certo microcomputador, o diálogo homem-máquina é realizado através de um teclado (input) e de um ecrã (output). Existem três rotinas que interessam de momento:

ecra, que coloca no ecrã (na posição referenciada por um cursor) o caracter cujo código ASCII se encontra no registo B e, além disto, faz avançar o cursor de uma posição.

teclado, que coloca no registo B o código ASCII da tecla actuada

linha, que coloca o cursor no início da próxima linha do ecrã.

Estas rotinas não alteram o conteúdo dos registos do microprocessador, salvo o caso da rotina **teclado**.

Apoiando-se nas rotinas indicadas, escrever um programa em Assembly, devidamente comentado, que lê do teclado dois caracteres e visualiza-os no ecrã separados por vírgula; se os dois caracteres lidos forem iguais, então aparecerá, na mesma linha, a mensagem IGUAIS, mas se forem diferentes a mensagem será DIFERENTES.
A execução do programa termina com a mensagem FIM quando, no teclado, se faz Z e Z. As mensagens IGUAIS, DIFERENTES e FIM são sempre escritas a partir da coluna 10.

Exemplo:

```
A,B      DIFERENTES
H,H      IGUAIS
T,8      DIFERENTES
Z,Z      FIM
```

18. Uma tabela de elementos (valores entre 0 e 255), sem elementos repetidos, está implementada em memória. A tabela inicia-se na posição de memória apontada por HL e o seu comprimento está no registo B.
Quando se chama uma rotina designada por INSERIR_CAR, um elemento que ainda não exista na tabela, colocado previamente no registo A, será inserido no final da tabela, havendo lugar à actualização do comprimento (conteúdo do registo B); se o elemento já existir na tabela, nada se altera.
Escrever em Assembly, devidamente comentada, a rotina INSERIR_CAR.

19. Supor agora uma lista de elementos (com valores entre 0 e 255) não repetidos, ordenados do menor para o maior, implementada em memória. A lista inicia-se na posição apontada por HL e o seu comprimento está no registo B.
A rotina PESQUI_ELEM procura na lista um elemento previamente colocado no registo A. Se encontra esse elemento, regressa com CARRY=1. Caso contrário, faz

CARRY=0 e HL apontará o primeiro elemento da lista cujo valor é superior ao elemento dado.
Escrever em linguagem Assembly e devidamente comentada a rotina PESQUI_ELEM.

20. Escrever em linguagem Assembly, bem comentada, a rotina INSERIR_ELEM, que insere na posição correcta da lista um novo elemento, previamente colocado no registo A. Esta rotina deverá utilizar PESQUI_ELEM.

21. Quando a rotina ROTINAX é chamada, o par HL aponta o início de um *buffer* (BUFF.DAD) que contem apenas valores positivos entre 0 e 255. A primeira posição deste *buffer* deve ser interpretada como sendo o seu comprimento. ROTINAX gera um segundo *buffer* (BUFF.RES) a partir da posição apontada por DE.

1ª posição de BUFF.RES – número de ocorrências do valor 0 em BUFF.DAD
2ª posição de BUFF.RES – número de ocorrências do valor 1 em BUFF.DAD
... ...

256ª posição de BUFF.RES – número de ocorrências do valor 255 em BUFF.DAD

No final da execução de ROTINAX deve ser reposto o valor inicial dos registos.

a) Apresentar um fluxograma que ilustre a estratégia que utilizar na implementação de ROTINAX.
b) Escrever a rotina ROTINAX, bem comentada, em Assembly Z80.
c) Escrever o programa principal que define a posição de BUFF.DAD e BUFF.RES e chama ROTINAX.

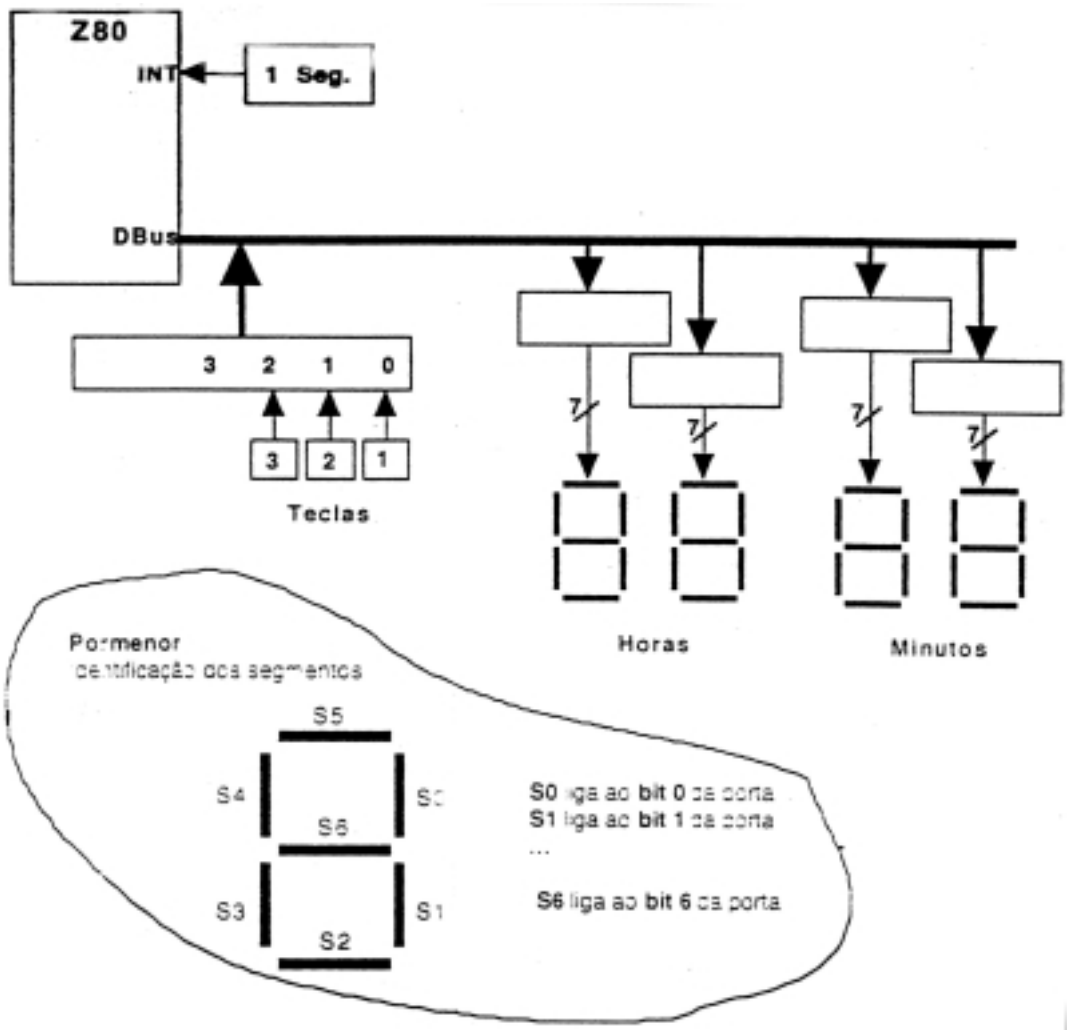
Exercícios sobre interrupções

22. Supor que INT (interrupt do Z80, **em modo 2**) é actuada por uma máquina, quando esta acaba de produzir uma peça. Para se desactivar INT é necessário enviar o valor 0FFh para a porta de saída de endereço 60h.
Escrever a rotina MAQUINA, devidamente comentada, que atende aquele pedido de interrupção, incrementando de +1 o par HL, registo que contem o número de peças produzidas até ao momento. Escrever também o programa principal, pondo em destaque tudo o que estiver relacionado com as inicializações das interrupções referidas.

23. Num microcomputador construído em torno do Z80 (ver figura), pretende-se implementar um relógio em tempo real, com visualização de **Horas** e **Minutos**.
É possível acertar o relógio através de um conjunto de três teclas.
As teclas são lidas por uma porta de entrada e os visualizadores de sete segmentos são controlados por quatro portas de saída.
Suponha-se que a linha de interrupção INT (programada em **modo 2**) é actuada de segundo a segundo, por uma base de tempo exterior.
Duas variáveis, designadas simbolicamente por **HORAS** e **MINUTOS**, contêm valores que variam de 0 a 23 e de 0 a 59, respectivamente.
Apresentar soluções, bem comentadas, para:

a) A rotina que **ATENDE** as interrupções a actualiza as variáveis HORAS e MINUTOS.

b) A rotina que **VISUALIZA** as Horas e Minutos, a partir do conteúdo das variáveis HORAS e MINUTOS.
c) A rotina que acerta o relógio, através das teclas 1, 2 e 3. Vejamos como funciona o acerto:
   - A tecla 3 faz o STOP/GO, pára ou faz funcionar o relógio.
   - Estando o relógio parado
     - a tecla 1 incrementa HORAS
     - a tecla 2 incrementa MINUTOS
   - estando o relógio a funcionar
     - as teclas 1 e 2 não são consideradas
d) o programa principal que, no essencial, controla as teclas, o visualizador e as interrupções, através das rotinas anteriormente referidas.



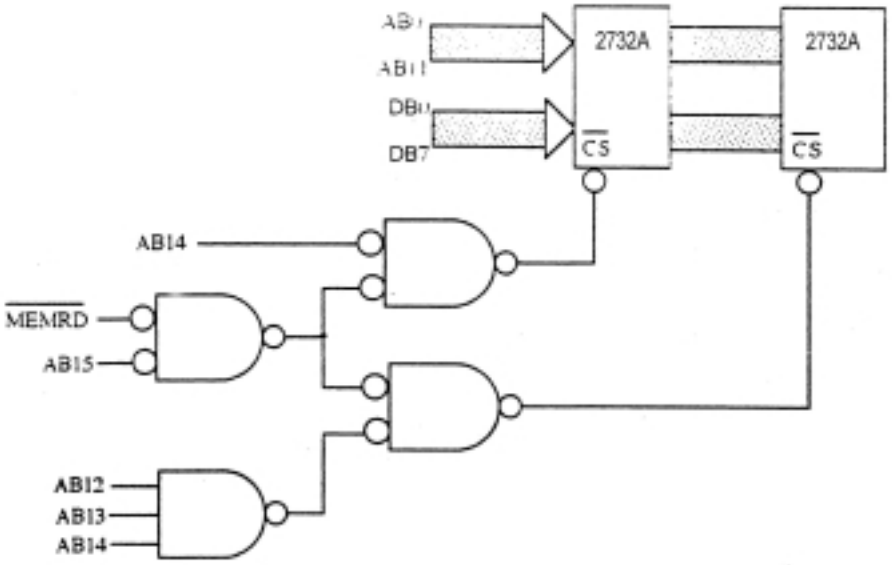
Exercícios sobre os barramentos do Microprocessador

24. A constituição da memória de um microcomputador é a seguinte:

- 24 K de RAM que começa em 0000h
- 12 K de ROM que termina em FFFFh.
- 8 K de EEPROM que começa em 8000h.

Desenhar o mapa de memória deste microcomputador.

25. Desenhar o mapa de memória para o sistema indicado na figura que se segue.



26. Para que endereços está activa a saída C da figura que se segue?