

寫下列題目中的組合語言程式可以使用的指令有：MOV, ADD, SUB, ADC, SBB, NEG, INC, DEC, JMP, JB, JNB, JA, JNA, JG, JNG, JL, JNL, JC, JO, JP, JS, JNC, JNS, JNO, JNS, JNP, JZ, JNZ, JCXZ, LOOP, LOOPZ, LOOPNZ, CALL, RET, PUSH, POP, AND, OR, NOT, XOR, TEST, CMP, STC, CLC 或課本第 6 章(包含)前，在課堂中講解過卻遺漏的其它指令。可以呼叫的作者提供的副程式，各題有不同的限制。

可能用到的 ASCII 碼：ENTER 是 0DH，換行是 0AH，數字'0'是 30H，英文字母'A'是 41H，'a'是 61H。

若覺得題目有錯誤，可說明錯誤處，並自行加以修改，而後解題。若題目無錯誤，而自行修改，當作是答錯。

1. 在資料段(data segment)中定義了一個字串資料，如下所示：

```
decout byte 10 dup (?)
```

其中，由 **decout** 起，已經放有幾位數正確的十進制數值，而以數碼 **OFFH** 當作結束。數碼 **OFFH** 以後的資料都不是我們要處理的。例如，內容依次是 1、2、3、4、**OFFH**、5、6、7、8 和 9，表示數值 1234，將被顯示在螢幕上。請寫出將這數值顯示到螢幕的組合語言程式部分。你只能呼叫教科書作者提供的字元輸出(WRITECHAR)副程式(程序)來寫作這個程式。

2. 資料段(data segment)中定義了一個字串資料 **str1** 和一個文字資料 **char1**，如下所示：

```
Str1 byte 100 dup (?)
```

```
Char1 byte ?
```

程式執行時，由位置 **str1** 起已放入一個以數值 0 結束的字串，**char1** 也已放入一個文字碼。請寫出在字串 **str1** 中找尋第一次出現存放在 **char1** 的文字的組合語言程式部分。若找到，暫存器 **ESI** 將存放它的位置，旗標 **CF** 設為 1；若找不到，**CF** 清為 0，**ESI** 的內容是 0。

3. 資料的定義如下所示：

```
SCORE BYTE 101 dup (?)
```

```
CLASS BYTE 5 dup (?)
```

在程式執行時，由位置 **SCORE** 起已放入某些(至多 100 個)8 位元(bit)正整數，代表一些考試分數。分數之後放著一個大於 100 的數值，表示其後(包含它)已沒有資料。現在想寫一個程式，將這些分數分類，計算各類的人數。大於或等於 90 分，是 A 類；大於或等於 80 分，小於 90 分，是 B 類；大於或等於 70 分，小於 80 分，是 C 類；大於或等於 60 分，小於 70 分，是 D 類；小於 60 分，是 E 類。各類別的人數依序存放在由 **CLASS** 起的記憶體位置中，順序是 A、B...、E，每類佔一個位元組(byte)。請寫出這部分的組合語言程式。

4. 我們想使用連續相加的方式來做整數的乘法。被乘數和乘數都是 8 位元的正整數，分別存放在記憶體中名為 **var1** 和 **var2** 的位置中，結果將存放在名為 **var3** 的 16 位元位置中。請寫一程式片斷，將 **var1** 連續相加 **var2** 次，以得到結果。注意：兩個 8 位元數值相乘，結果

可能需 16 位元來存放。

5. 請說明下列程式片斷執行每一個指令後，暫存器(EAX、EBX、ECX、ESI 或 EDI)或記憶體位置的內容有甚麼改變。假設在執行第一個指令前，暫存器 EAX 的內容是 12345678H，EBX 的內容是 0ABCDEF12H，ECX 的內容是 10203043H，ESI 和 EDI 的內容都是 0。你的答案可以是個數值，如 12H；也可以是個未知值(地址或其它定義常數)，如 VAR1+4。記憶體內容可寫成記憶體位置 VAR3 存 32 位元的 12345678H，或記憶體位置 VAR3+1 存 8 位元的 12H。

```
.DATA
```

```
VAR1  DWORD  1
```

```
VAR2  DWORD  2
```

```
VAR3  DWORD  3
```

```
.CODE
```

```
MOV    ESI,OFFSET VAR1
```

```
MOV    EDI,OFFSET VAR2
```

```
MOV    [ESI],EAX
```

```
MOV    [EDI],EBX
```

```
INC    ESI
```

```
DEC    EDI
```

```
MOV    EAX,[EDI]
```

```
MOV    EBX,[ESI]
```

```
AND    AX,0FFH
```

```
MOVSX  ECX,BYTE PTR [EDI+2]
```