

1)

```
ORG 1000H
NUM1  DW  9234H  }
        DW  1111H  } NUM1 = 11119234H
NUM2  DW  8432H  }
        DW  2222H  } NUM1 = 22228432H
RESUL  DW  ?
        DW  ?
BIEN   DB  0
MAL    DB  0
```

```
ORG 2000H
MOV  AX, NUM1 ;  AX:=9234H  Parte baja de NUM1
MOV  CX, NUM2 ;  CX:=8432H  Parte baja de NUM2
MOV  DX, NUM1+2; DX:=1111H  Parte alta de NUM1
MOV  BX, NUM2+2; CX:=2222H  Parte alta de NUM2
ADD  AX, CX ;    AX:=AX+BX  Suma parte baja
ADC  DX, BX ;  DX:=DX+BX+Carry ( Carry generado por la suma de la parte baja )
JO   resul_mal
MOV  BIEN, 0FFH
JMP  fin
resul_mal: MOV MAL, 0FFH
fin:   MOV  RESUL, AX
        MOV  RESUL+2, DX
        HLT
        END
```

El resultado de la suma queda almacenado en DX AX. (32 bits). Aquí consideramos que la parte alta de la suma no genera Carry.

Para ver si es correcto el resultado en Ca2, debemos verificar la ausencia de overflow. Después de la suma escribimos lo siguiente:

```
JO   resul_mal ; Checkea el bit de overflow. Salta si O=1 (resultado ca2 mal)
MOV  BIEN, 0FFH ; No saltó. Resultado bien en ca2.
JMP  fin ;
resul_mal: MOV MAL, 0FFH
fin:
```

2) Los arreglos tienen definido sólo un elemento de 32 bits y no 6 como dice el enunciado.

	ORG 1000H		
TAB1	DW 11B2H	}	El número es 234511B2H
	DW 2345H		
TAB2	DW 0061H	}	El número es 21F00061H
	DW 21F0H		
TAB3	DW 0000H	}	La suma es 45351213H
	DW 0000H		

```
ORG 2000H
MOV DX, 0 ; Almacena el carry
MOV CX, 0 ; Índice del lazo
SIGO: MOV BX, OFFSET TAB1 ; Dirección de comienzo de TAB1
      ADD BX, CX ; Apunta a una determina word en TAB1
      MOV AX, [BX]
      ADD AX, DX ; Sumó el carry anterior
      MOV DX, 0 ; Pone a 0 para guardar el nuevo carry
      MOV BX, OFFSET TAB2 ; BX:= Dirección de TAB2
      ADD BX, CX ; Apunta a una determinada word en TAB2
      ADD AX, [BX] ; Suma dos words
      ADC DX, 0 ; DX:= DX + 0 + Carry
      MOV BX, OFFSET TAB3 , Dirección de TAB3
      ADD BX, CX ; Apunta a una determinada palabra en TAB3
      MOV [BX], AX ; Amacena el resultado en TAB3
      ADD CX, 2 ; Apunta a la siguiente word
      CMP CX, 4
      JNZ SIGO
      HLT
      END
```

MEMORIA	
0FFF	00
1000	B2
1001	11
1002	45
1003	23
1004	61
1005	00
1006	F0
1007	21

TAB1

TAB2

MEMORIA	
1FFF	00
2000	BA
2001	00
2002	00
2003	B9
2004	00
2005	00
2006	BB
2007	00

Acá comienza
el programa

MEMORIA	
1004	61
1005	00
1006	F0
1007	21
1008	13
1009	12
100A	35
100B	45
100C	00

TAB3

Contenido en
memoria después
de realizar la suma

6)

```
                ORG 1000H
MANTISA  DB  0FH (00001111b)
EXPONENTE DB  08H (00001000b)
```

```
                ORG 2000H
MOV  AH, MANTISA
MOV  AL, EXPONENTE
CALL AJUSTE
HLT
END
```

; SIN VERIFICAR

```
                ORG 3000H
AJUSTE:  DEC AL
        ADD AH, AH
        RET
```

;CON VERIFICACIÓN

```
                ORG 3000H
AJUSTE:  MOV CL, AH
        AND CL, 10000000b
        JNZ NOSEPUDO
        DEC AL
        ADD AH, AH
        MOV BL, 00H
        JMP FIN
NOSEPUDO: MOV BL, 0FFH
        FIN:  RET
```

7)

```
                                ORG 3000H
FLOTANTE:  MOV CX, AX
                                ADD CX, CX ; Exponente en CH, le sacamos el signo
                                CMP CH, 0
                                JNZ VER_UNOS ; Si no salta E=0
                                MOV CL, 0 ; CL indicación E=0
                                JMP SIGO
VER_UNOS:  CMP CH, 11111111b ; FFH
                                JNZ FIN ; No es ningún caso especial
                                MOV CL, 1 ; CL indicación E=255 (todos unos)
;
; Vamos a ver cuanto vale la mantisa
;
                                SIGO:  AND AL, 01111111b ; (7FH)
                                CMP AL, 0 ; Primera parte de la mantisa es 0
                                JNZ MANTI_DIS_CERO
                                CMP BX, 0 ; Segunda parte de la mantisa es 0
                                JNZ MANTI_DIS_CERO
                                MOV CH, 0 ; CH indicación M=0
                                JMP SIGO_1
MANTI_DIS_CERO:  MOV CH, 1 ; CH indicación M distinta de 0
;
; Veamos M y E que valores tienen
;
                                SIGO_1:  MOV AH, CH
                                MOV AL, CL
                                AND AL, 1
                                JZ EXPO_CERO
                                AND AH, 1 ; E = 255
                                JZ INFINITO
                                MOV BL, 2 ; Número es NAN
                                JMP FIN_1
EXPO_CERO:  AND CH, 1
                                JZ CERO ; Si salta el número es 0
                                MOV BL, 0 ; Número es denormalizado
                                JMP FIN_1
INFINITO:   MOV BL, 1 ; Número es infinito
                                JMP FIN_1
CERO:      MOV BL, 3 ; Número es 0
                                JMP FIN_1
FIN:       MOV BL, 4 ; Número está normalizado
FIN_1:     RET
```