Universidad
Francisco de Vitoria
UFV Madrid

Fundamentos de la ingeniería informática

Ingeniería de sistemas industriales

Curso 2019-2020

Sumador

1. Enunciado

ov es un byte en la dirección $1000,

a, b y c son enteros con signo de 64 bits Big Endian, situados a continuación de ov.

Se pide escribir un programa en ensamblador de CH-2020 que realice la siguiente operación: C=A+B; en caso de que la operación produzca overflow: ov=$FF si no ov=$00

2. Con variables globales

El problema que nos plantea este ejercicio es la necesidad de manejar tres vectores mientras que tan sólo tenemos dos registros de direcciones. Para resolverlo vamos a utilizar punteros (variables que almacenan direcciones). Reservemos espacio para esas tres variables.

```
variables = 1000

puntero_A       space 2         ; puntero al vector A
puntero_B       space 2         ; puntero al vector B
puntero_C       space 2         ; puntero al vector C
contador        space 1         ; contador
flags           space 1         ; para almacenar temporalmente el registro F (CZS)
```

La variable contador la necesitaremos para contar las 8 repeticiones que necesitaremos para realizar la suma completa.

Necesitaremos también definir también alguna constante

```
LONGITUD        = 8             ;La longitud de los vectores
```

Otro problema que plantea este ejercicio es que el almacenamiento de datos en el entero con signo de 64bits es big endian, como la operación de suma se realiza de menos significativo a más significativo es necesario recorrer los operandos de final a principio. Para calcular el final de un array tan sólo hemos de sumar LONGITUD-1 (¿por qué?) a su dirección de inicio; para ello ya hemos desarrollado en clase la función incrementar.

```
        MOV     MSB(vector_a) H0
        MOV     LSB(vector_a) L0
        MOV     #LONGITUD D0
        CALL    Incrementar
        DEC     A0
```

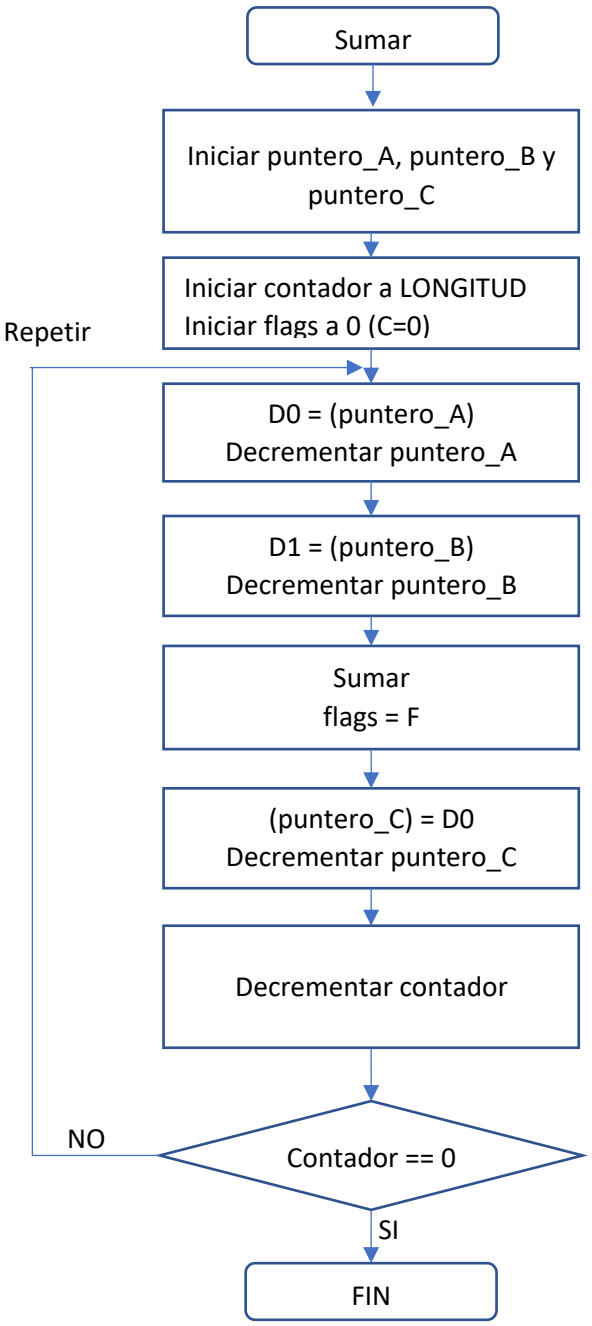
Este resultado necesitaremos guardarlo en la variable que hemos creado puntero_A (lo mismo con B y C)

Otro elemento de complejidad es que en puntero_X tenemos la dirección del dato, pero para acceder a esa dirección hemos de hacerlo de forma indirecta, lo que implica que hay que llevar la dirección de puntero_X a un registro de direcciones (byte a byte), después el contenido de esa dirección (que es otra dirección) al otro registro de direcciones y ahora sí ese registro apunta al dato.

```
;D0 = (puntero_A) y Decrementar puntero_A
mov     #MSB(puntero_A) H1          ; Dirección del puntero_A al A1
mov     #LSB(puntero_A) L1
mov      (A1) H0                    ; Contenido puntero_A al A0
inc     A1
mov      (A1) L0

mov     (A0) D0                     ; Contenido de la posición actual del vector a D0
```

Con estas piezas podemos ponernos a pensar en un organigrama que resuelva el problema.


Sumar
Iniciar puntero_A, puntero_B y puntero_C
Iniciar contador a LONGITUD
Iniciar flags a 0 (C=0)
Repetir
D0 = (puntero_A)
Decrementar puntero_A
D1 = (puntero_B)
Decrementar puntero_B
Sumar
flags = F
(puntero_C) = D0
Decrementar puntero_C
Decrementar contador
NO
Contador == 0
SI
FIN

Y ahora pasamos a programarlo

```
;Constantes
LONGITUD = 8

;Espacio para los datos
Datos = 1000
sumando_A = space LONGITUD
Sumando_B = space LONGITUD
Sumando_C = space LONGITUD

;Espacio para las variables auxiliares que se necesitan
variables = 1100
puntero_A       space 2         ; puntero al vector A
puntero_B       space 2         ; puntero al vector B
puntero_C       space 2         ; puntero al vector C
contador        space 1         ; contador de repeticiones
flags           space 1         ; para almacenar temporalmente el registro F (CZS)

Reset   = 0
        JMP     Inicio

Inicio  = 100
        ; Iniciar puntero_A
        mov     #MSB(sumando_A) H0          ; Dirección del sumando_A al A0
        mov     #LSB(sumando_A) L0
        mov     #LONGITUD D0                ; Calcular el final del sumando A
        call    Incrementar
        dec     A0
        mov     #MSB(puntero_A) H1          ; Dirección del puntero_A al A1
        mov      #LSB(puntero_A) L1
        mov     H0 (A1)                     ; Dirección del sumando_A al puntero_A
        inc     A1
        mov     L0 (A1)

        ; Iniciar puntero_B
        mov     #MSB(sumando_B) H0          ; Dirección del sumando_B al A0
        mov     #LSB(sumando_B) L0
        mov     #LONGITUD D0                ; Calcular el final del sumando B
        call    Incrementar
        dec     A0
        mov     #MSB(puntero_B) H1          ; Dirección del puntero_B al A1
        mov     #LSB(puntero_B) L1
        mov     H0 (A1)                     ; Dirección del sumando_B al puntero_B
        inc     A1
        mov     L0 (A1)

        ; Iniciar puntero_C
        mov     #MSB(sumando_C) H0          ; Dirección del sumando_C al A0
        mov     #LSB(sumando_C) L0
        mov #LONGITUD D0                    ; Calcular el final del sumando C
        call Incrementar
```

```
        dec     A0
        mov     #MSB(puntero_C) H1          ; Dirección del puntero_C al A1
        mov     #LSB(puntero_C) L1
        mov     H0 (A1)                     ; Dirección del final del sumando_C al puntero_C
        inc     A1
        mov     L0 (A1)

        ; Iniciar contador a LONGITUD
        mov     #LONGITUD D0
        mov     D0 contador

        ; Iniciar flags a 0
        mov     #0 D0
        mov     D0 flags

Repetir
        ;D0 = (puntero_A) y decrementar puntero_A
        mov     #MSB(puntero_A) H1          ; Dirección del puntero_A al A1
        mov     #LSB(puntero_A) L1
        mov      (A1) H0                    ; Contenido puntero_A al A0
        inc     A1
        mov      (A1) L0

        mov     (A0) D0                     ; Contenido de la posición actual del vector a D0

        dec     A0                          ; Decrementar
        mov     L0 (A1)                     ; Guardar la nueva dirección en el puntero_A
        dec     A1
        mov     H0 (A1)

        ;D1 = (puntero_B) y decrementar puntero_B
        mov     #MSB(puntero_B) H1          ; Dirección del puntero_B al A1
        mov     #LSB(puntero_B) L1
        mov      (A1) H0                    ; Contenido puntero_B al A0
        inc     A1
        mov      (A1) L0

        mov     (A0) D1                     ; Contenido de la posición actual del vector a D1

        dec     A0                          ; Decrementar
        mov      L0 (A1)                    ; Guardar la nueva dirección en el puntero_B
        dec     A1
        mov     H0 (A1)

        ;Sumar
        mov     flags F                     ; recuperar el estado del carry
        ADD
        mov     F flags                     ; salvar el nuevo estado del carry

        ;(puntero_C) = D0 y decrementar puntero_C
        mov     #MSB(puntero_C) H1          ; Dirección del puntero_B al A1
        mov     #LSB(puntero_C) L1
```

```
        mov     (A1) H0                 ; Contenido puntero_B al A0
        inc     A1
        mov     (A1) L0

        mov     D0 (A0)                 ; Contenido de la posición actual del vector a D1

        dec     A0                      ; Decrementar
        mov     L0 (A1)                 ; Guardar la nueva dirección en el puntero_B
        dec     A1
        mov     H0 (A1)

        ;Decrementar contador
        mov     contador D0
        mov     #1 D1
        clc
        sub
        mov     D0 contador

        ;¿Contador == 0?
        JR      NZ repetir

Fin
        STOP

Incrementar
        clc
        mov     L0 D1
        add
        mov     D0 L0
        mov     #0 D0
        mov     H0 D1
        add
        mov     D0 H0

        ret
```

3. Una mejora

Si te fijas el código

```
        mov     (A1) H0
        inc     A1
        mov     (A1) L0

        mov     (A0) D0

        dec     A0
        mov     L0 (A1)
        dec     A1
        mov     H0 (A1)
```

se repite dos veces, esto es un síntoma que puede apuntar a que sería bueno convertirlo en una subrutina. Ésta toma A1 como la dirección de un puntero, toma lo apuntado por dicho puntero e incrementa el puntero.

Podemos definir que la subrutina recibe como entrada A1 y devuelve D0 = ((A1)); además de decrementar el puntero, (A1) = (A1)-1. Debemos anotar también que el A0 y A1 quedan modificado s en la operación.

Si llamamos Leer_y_retroceder a la rutina:

```
Leer_y_retorceder
        mov     (A1) H0             ; Obtener la dirección origen del dato
        inc     A1
        mov     (A1) L0

        mov     (A0) D0             ; Leer el dato

        dec     A0                  ; Decrementar el puntero

        mov     L0 (A1)             ; Reponer el puntero en memoria
        dec     A1
        mov     H0 (A1)

        ret
```

Aunque no se repite el código que realiza la función contraria también podemos convertirlo en una subrutina lo que simplificará la lectura del código (otro objetivo/síntoma para la creación de subrutinas). Las subrutinas favorecen la encapsulación de operaciones ocultando los detalles detrás de una etiqueta; si seleccionas una etiqueta esta indicará **qué** hace obviando el **cómo**.

La función Escribir_y_retroceder recibe como entrada A1 dirección del puntero y D0 el dato a escribir en la memoria apuntada. Realiza ((A1)) = D0 y (A1) = (A1)-1. El registro A0 y A1 quedan modificados en la operación.

```
Escribir_y_retorceder
        mov     (A1) H0         ; Obtener el puntero destino
        inc     A1
        mov     (A1) L0

        mov     D0 (A0)         ; Leer el dato

        dec     A0              ; Decrementar el puntero

        mov     L0 (A1)         ; Reponer (salvar) el puntero
        dec     A1
        mov     H0 (A1)

        ret
```

Podemos ser algo más radicales y convertir la iniciación de cada puntero en una subrutina

```
Iniciar_puntero_A
        mov     #MSB(sumando_A) H0          ; Dirección del sumando_A al A0
        mov     #LSB(sumando_A) L0
        mov     #LONGITUD D0                ; Calcular el final del sumando A
        call    Incrementar
        dec     A0
        mov     #MSB(puntero_A) H1          ; Dirección del puntero_A al A1
        mov     #LSB(puntero_A) L1
        mov     H0 (A1)                     ; Dirección del sumando_A al puntero_A
        inc     A1
        mov     L0 (A1)

        ret

Iniciar_puntero_B
        mov     #MSB(sumando_B) H0          ; Dirección del sumando_B al A0
        mov     #LSB(sumando_B) L0
        mov     #LONGITUD D0                ; Calcular el final del sumando B
        call    Incrementar
        dec     A0
        mov     #MSB(puntero_B) H1          ; Dirección del puntero_B al A1
        mov     #LSB(puntero_B) L1
        mov     H0 (A1)                     ; Dirección del sumando_B al puntero_B
        inc     A1
        mov     L0 (A1)

        ret

Iniciar_puntero_C
        mov     #MSB(sumando_C) H0          ; Dirección del sumando_C al A0
```

```
        mov     #LSB(sumando_C) L0
        mov #LONGITUD D0                ; Calcular el final del sumando C
        call Incrementar
        dec     A0
        mov     #MSB(puntero_C) H1      ; Dirección del puntero_C al A1
        mov     #LSB(puntero_C) L1
        mov     H0 (A1)                 ; Dirección del final del sumando_C al puntero_C
        inc     A1
        mov     L0 (A1)

        ret
```

Simplemente hemos convertido los comentarios, que parecían aceptables, en etiquetas de subrutinas y añadido el correspondiente ret.

El código que nos queda:

```
;Constantes
LONGITUD = 8

;Espacio para los datos
Datos = 1000
sumando_A = space LONGITUD
Sumando_B = space LONGITUD
Sumando_C = space LONGITUD

;Espacio para las variables auxiliares que se necesitan
variables = 1100
puntero_A       space 2         ; puntero al vector A
puntero_B       space 2         ; puntero al vector B
puntero_C       space 2         ; puntero al vector C
contador        space 1         ; contador de repeticiones
flags           space 1         ; para almacenar temporalmente el registro F (CZS)

Reset   = 0
        JMP     Inicio

Inicio  = 100
        call    Iniciar_puntero_A
        call    Iniciar_puntero_B
        call    Iniciar_puntero_C

        ; Iniciar contador a LONGITUD
        mov     #LONGITUD D0
        mov     D0 contador

        ; Iniciar flags a 0
        mov     #0 D0
        mov     D0 flags

Repetir
        ;D1 = (puntero_A) y decrementar puntero_A  (he cambiado el registro destino)
```

```
        mov     #MSB(puntero_A) H1          ; Dirección del puntero_A al A1
        mov     #LSB(puntero_A) L1
        call    Leer_y_retroceder

        mov     D0 D1           ;Ojo este mov es necesario ya que en el siguiente bloque
                                ; se vuelve a invocar a Leer_y_retroceder que modifica D0
                                ; pero no D1

        ;D0 = (puntero_B) y decrementar puntero_B  (aquí he vuelto a cambiar el registro destino)
        mov     #MSB(puntero_B) H1          ; Dirección del puntero_B al A1
        mov     #LSB(puntero_B) L1
        call    Leer_y_retroceder

        ;Sumar
        mov     flags F                     ; recuperar el estado del carry
        ADD
        mov     F flags                     ; salvar el nuevo estado del carry

        ;(puntero_C) = D0 y decrementar puntero_C
        mov     #MSB(puntero_C) H1          ; Dirección del puntero_B al A1
        mov     #LSB(puntero_C) L1
        call    Escribir_y_retroceder

        ;Decrementar contador
        mov     contador D0
        mov     #1 D1
        clc
        sub
        mov     D0 contador

        ;¿Contador == 0?
        JR      NZ repetir

Fin
        STOP
;*****************************************************************
; Espacio para las subrutinas
;*****************************************************************
Leer_y_retorceder
        mov     (A1) H0             ; Obtener la dirección origen del dato
        inc     A1
        mov     (A1) L0

        mov     (A0) D0             ; Leer el dato

        dec     A0                  ; Decrementar el puntero

        mov     L0 (A1)             ; Reponer el puntero en memoria
        dec     A1
        mov     H0 (A1)

        ret
```

```
Escribir_y_retorceder
        mov     (A1) H0         ; Obtener el puntero destino
        inc     A1
        mov     (A1) L0

        mov     D0 (A0)         ; Leer el dato

        dec     A0              ; Decrementar el puntero

        mov     L0 (A1)         ; Reponer (salvar) el puntero
        dec     A1
        mov     H0 (A1)

        ret
Iniciar_puntero_A
        mov     #MSB(sumando_A) H0          ; Dirección del sumando_A al A0
        mov     #LSB(sumando_A) L0
        mov     #LONGITUD D0                ; Calcular el final del sumando A
        call    Incrementar
        dec     A0
        mov     #MSB(puntero_A) H1          ; Dirección del puntero_A al A1
        mov     #LSB(puntero_A) L1
        mov     H0 (A1)                     ; Dirección del sumando_A al puntero_A
        inc     A1
        mov     L0 (A1)

        ret

Iniciar_puntero_B
        mov     #MSB(sumando_B) H0          ; Dirección del sumando_B al A0
        mov     #LSB(sumando_B) L0
        mov     #LONGITUD D0                ; Calcular el final del sumando B
        call    Incrementar
        dec     A0
        mov     #MSB(puntero_B) H1          ; Dirección del puntero_B al A1
        mov     #LSB(puntero_B) L1
        mov     H0 (A1)                     ; Dirección del sumando_B al puntero_B
        inc     A1
        mov     L0 (A1)

        ret

Iniciar_puntero_C
        mov     #MSB(sumando_C) H0          ; Dirección del sumando_C al A0
        mov     #LSB(sumando_C) L0
        mov #LONGITUD D0                    ; Calcular el final del sumando C
        call Incrementar
        dec     A0
        mov     #MSB(puntero_C) H1          ; Dirección del puntero_C al A1
        mov     #LSB(puntero_C) L1
        mov     H0 (A1)                     ; Dirección del final del sumando_C al puntero_C
        inc     A1
```

```
        mov     L0 (A1)

        ret

Incrementar
        clc
        mov     L0 D1
        add
        mov     D0 L0
        mov     #0 D0
        mov     H0 D1
        add
        mov     D0 H0
        ret
```

El programa principal ha quedado bastante reducido, pero sobre todo simplificado, porque hemos trasladado la complejidad y el detalle a las subrutinas.