

# Assembler

Un programa en Assembler AFC declara un arreglo de 20 elementos enteros signados y carga en él los primeros 20 valores negativos leídos de un dispositivo de entrada mapeado en A0010124h.

```

.begin
.org 2048
Array: .dwb 20      ! Array(20): entero signado
Direccion .equ A0010124h ! dirección del perif.
Add %r0, %r0, %r1      ! índice del arreglo → r1, lo seteo en 0.
Sub %r1, 80, %r0      ! While cuenta Negativos < 20 do
be EndWhile             ! branch si %r1 - 80 == 0
sehi 280040h, %r8      ! 101000000000000010000000 = 280040h son
                        ! los 22 bits más significativos
Or  %r8, A0124h, %r8    ! 0100100100 = 0124h 10 bits menos sian
Id  %r8, %r2             ! r2
add %r0, %r2, %r0        ! if
bpos While              !
st  %r2, [Array], %r1    ! A
add %r1, 4, %r1          !
ba  While                !
endwhile:
Jmpl  %r15, 4, %r0       ! retv
.end

```

el mismo enunciado pero implementando una subrutina de un dispositivo de un periférico. Esta subrutina recibe por stack como parámetro de entrada la dirección del periférico y devuelve por stack el dato leído.

Declaración de la subrutina

Invocación de la subrutina

Leeperif: Id %r14, %r11      lru la dir en fx: contiene la dirección de la fila  
                Id %r11, %r10

St %r10, %r14      en r14 dejo el contenido  
Jmpl %r15, 4, %r0

↓  
la pos  
expandida  
en r14

St %rX, %r14

Call Leeperif

Id %r14, %r15

↓

# Assembler

Un Programa en Assembler ABC declara un arreglo de 20 elementos enteros signados y carga en él los primeros 20 valores negativos leídos de un dispositivo de entrada mapeado en A0010124h

```
.begin
```

```
.org 2048
```

Array: .dw 20 ! Array(20): entero signado

Direccion .equ A0010124h ! dirección del Perif.

Add %r0, %r0, %r1 ! índice del arreglo → r1, lo seteo en 0.

Sub %r1, 80, %r0 ! While cuenta Negativos < 20 do

be end While ! branch si %r1 - 80 == 0

sehi 280040h, %r8 ! 101000000000000010000000 = 280040h son  
! los 22 bits más significativos

Or %r8, 0124h, %r8 ! 0100100100 = 0124h 10 bits menos sign.

Id %r8, %r2 ! r2 → Valor leído

Add %r0, %r2, %r0 ! if lectura < 0

bpos While

St %r2, [Array], %r1 ! Arreglo(indice) = 1

Add %r1, 4, %r1

ba While

endwhile:

Jmpf %r15, 4, %r0 ! return

end

el mismo enunciado pero implementando una subrutina dedicada a leer datos de un periferico. Esta subrutina recibe por stack como parámetro de entrada la dirección del periférico y devuelve por stack el dato leído.

Declaración de la subrutina

Invocación de la subrutina

Leerperif: Id %r14, %r11 ! la 1a dir en r14 contiene la dirección  
Id %r11, %r10 ! de la pila :

St %r10, %r14 en r14 desd el contenido

Jmpf %r15, 4, %r0

St %rX, %r14

Call Leerperif

Id %r14, %r15

La pos  
apuntador  
r14

.begin

org 2048

Array: dwb 20

Direccion .equ A0010124h

add %10, %10, %11

while : sub %11, 80, %10

be endWhile

sehi....

I cargo  
sguicel

add %114, -4, %114

st %18, %114

call LeerPerif

ld %114, %12

add %114, 4, %114

add %10, %12, %10 !if lectura. L0

bpos while

st %12, [Array], %11

add %11, 4, %11 !indice = indice + 1

ba While

JMP1 %115, 4, %10

LeerPerif: ld %114, %11

ld %11, %10

st %10, %114

JMP1 %115, 4, %10

.end

Un periférico está mapeado en C21000A1h entrega datos de 32 bits en formato de punto flotante simple precisión. Escribir un programa que lee 64 valores de ese periférico y devuelva por stack la cantidad de valores positivos entregados por el mismo. La lectura de cada dato del periférico debe ser realizada por una subrutina que no recibe parámetros y que devuelve vía stack el valor leído. Escribir en el mismo módulo el programa ppal y la subrutina. (12-02-20)

```

.begin
.org 2048
    SP .equ %r14           ! llamo sp al puntero a la pila
    ADDCC %r0, 64, %r1       ! r1 → # veces a leer.
    Loop: ANDCC %r1, %r1, %r0
    BE FIN
    ADDCC %r1, -1, %r1
    CALL lectura             ! lectura → subrutina
    LD SP, %r3                ! lo que está en la pila lo cargo en r3
    ADDCC SP, 4, SP
    ADDCC %r3, %r0, %r3
    breq loop                 ! si es negativo sigo
    ADDCC %r5, 1, %r5         ! si no es le sumo uno al contador.
    BA LOOP
    lectura: SEIHI C21000h, %r2
    SII %r2, 2, %r2
    ADDCC %r2, 0A1h, %r2
    ADDCC SP, -4, SP
    ST %r2, SP
    JMPI %r15+4, %r0
    ADDCC SP, -4, SP
    ST %r5, SP
    Fin JMPI %r15+4, %r0
.end

```

Un programa lee números entregados por un dispositivo de entrada y guarda en dos arreglos según si el número >= 0. Ambos arreglos de 48 elementos cada uno, son declarados por el mismo programa. Una y otra vez se repite el proceso hasta que se agota la capacidad de alguno. Una rutina declarada en otro módulo tiene la función de leer 5 valores del periférico (mapeado en C0102030h) y devolver el valor vía stack (24/07/18)

.begin

.ORG 2048

.EXTERN lec5

.macro PUSH ARG

ADD %R14,-4,%R14

ST ARG,%R14

.endmacro

.macro POP ARG

LD %R14,ARG

! macro ins para agregar elem a la pila

ADD %R14,4,%R14

! macro ins para obtener elem de la pila

.endmacro

.macro RONER DIR, VONC

SUB DIR,CONT,%R8

ST %R5,%R8

ADD CONT,-4,CONT

.endmacro

DIR1 .EQU 3000+192

! vectores para los numeros negativos/positivos

DIR2 .EQU 3192+192

!

main: OR %R0,DIR1,%R1

! R1 → puntero a arreglo pos

OR %R0,DIR2,%R2

! R2 → puntero a vec negs

SEZHI 294028h,%R20

! dir del

ADD %R20,120h,%R20

! periferico.

OR %R0,192,%R3

! Tamaños de

OR %R0,192,%R4

! los arreglos

PUSH %R15

loop: ANDCC %R3,%R3,%R0

! Si algun vector se que da sin

be end

! Capacidad termina

ANDCC %R4,%R4,%R0

be end

Call leer

POP %15 ! guarda el valor leido en %15

Adddec %15, %\$5, %10 ! veo si 15 es + o -

breq negativos

Push %11, %13

ba loop

leer: ld %120, %15 ! cargo en %13 el 1º elem del vector

Push %15

JMP1 %115, 4, %10

end: POP %115

JMP1 %115, 4, %10

.org 3000

Arreglo 1: .dw 48

Arreglo 2: .dw 48

.end

Un programa recibe por stack la dir de inicio y el largo de un array. Si arr[0] tiene en los 2 bits más significativos, lo reemplaza por 0, en caso que no altere ningún elemento en el último elemento carga 0x00000000h, la condición es evaluada por una subrutina a la que se le pasa por stack el valor a evaluar y devuelve 1 si se verifica la cond y debe ser declarada en el mismo módulo. (30-07-2018) (hay q' definir push y pop igual q' siempre, no lo incluyo como dato)

begin

.org 2048

main: ld[r1], %r1

!r1 Puntero

ld[r2], %r2

!r2 → Tamaño

push %r1

push %r2

pop %r2

pop %r1

and %r0, %r0, %r5 ! se reemplaza 0 a r5

add %r1, %r2, %r1 ! r1 al final del arreglo.

loop: andcc %r2, %r2, %r0

be fin\_ciclo

sub %r1, %r2, %r3 ! r3 → dirección del dato

ld %r3, %r4 ! r4 → dato a evaluar

push %r4

call evaluar

pop %r4 ! r4 → el return de evaluar (1/0)

add %r5, %r4, %r5 ! r5+=r4

andcc %r4, %r4, %r0

be sigo ! si la función devolvió 0 sigo

si %r0, %r3 ! si no, cargo un 0 en %r3

sigo: add %r2, -4, %r2 ! avanzar pila

ba loop

fin\_ciclo: andcc %r5, %r5, %r0

be bandera ! si r5 es 0, no pise ningún dato

ba end

bandera: sethi 34400000h, %r6

add %r6, 1600h, %r6

Add %r1, -4, %r1 ! r1 → dir ultimo elemento

st %r0, %r1

ba end

evaluar: pop %r7

srl %r7, 30, %r7 ! en r7 me quedo con los 2 bits mas significativos y lo

subcc %r7, 3, %r0

be Cambio

and %r0, %r0, %r8 ! seteo a r8 en 0

ba return

Cambio: or %r0, 1, %r8 ! r8=1

return: push %r8

jmp l %r15, 4, %r0

dir: 3000

lom: 16

end: wait

.end

setear: srl %r1, 30, %r5

subcc %r5, 3, %r0

be or %r1, %r0, %r1

add %r6, 1, %r6 ! r6 contador

Un Programa declara un arreglo de 32 palabras de 32 bits y carga en el 10.5 primeras 32 lecturas de un periférico que está mapeado en la dir C3101200h. Una vez finalizada esta tarea invoca una subrutina que calcula su promedio. El programa final devuelve el promedio por stack y termina. Valores Periférico → ref en complemento a 2. La rutina debe ser declarada en el mismo módulo que el programa final, recibe por stack la dir del arreglo y devuelve por stack el promedio o 0 si se excede (18-02-2020)

.begin

.org 2048

%SP . equ 1514

! llamo sp al stack pointer

addcc 15, 10, 16

! guardar la ref de r15 antes de llamar a una subr

Comienzo\_arr .dwb 32 2052

sethi 30C404h, r1

! cargo el perif en un reg

add 200h, r1

!

addcc 10, 2052, 12

! el comienzo del arreglo

addcc 10, 128, r3

! cantidad de lecturas (32 palabras \* 4 bits)

loop: addcc r3, r3, 10

be fin

addcc r3, -4, r3

! recorre la pila

ld r1, r5

! lo que está en el periférico lo cargo en r5

addcc r2, r3, r20

! guardo en r20 la pos i del arreglo

st r5, r20

! en la pos i guardo el valor leído del perif

ba loop

fin: call promedio

Promedio: addcc r0, r0, r21

! aux inicializado en 0

addcc r0, 128, r3

! cantidad de lecturas

loop2: addcc r3, r3, r0

be fin2

addcc r3, -4, r3

ld r2, r3, r22 ! guarda comienzo + i en r22

addcc r21, r22, r21 ! se lo sumo al auxiliar (acumulador)

bvs desbord

ba loop → div 2 (1) → div 4 (2) veces

fin2:  $\text{SHL } \%s21, 5, \%s21$  ! dividido por 32 (mover n pos es  $/2^n$ )

JMP1  $\%s15+4, \%10$

desborde:  $\text{id0 in t0pia} \rightarrow \text{ST } \%s21, \text{SP}$

JMP1 ..

end

12-12-2019

a)  
.begin  
.org 2043

.macro push org  
add %r14, -4, %r14  
st org, %r14

.endmacro

.macro ref org  
ld %r14, org  
add %r14, 4, %r14  
.endmacro

DIR .equ 3000

add %r15, %r0, %r16 ! se guarda la ref a r15 en r16

add %r0, DIR, %r1 ! Puntero al arreglo

sehi d562d1h, %r20

sll %r20, 2, %r20

add %r20, 120h, %r20 ! cargo el perif.

add %r0, 20, %r2 ! r2 long del arreglo

loop: andcc %r2, %r2, %r0

be end

call leer

pop %r5

! guarda en r5 lo que retorna la rutina

sc %r5, %r1

! guarda en el array el res.

add %r1, 4, %r1

! avanza array.

sub %r2, 1, %r2

ba sleep

salida: jmel %r16, 4, %r0

leer: ld %r20, %r3

! cargo lo leido en r3

sll %r3, 14, %r3

! saco los bits que no

sra %r3, 14, %r3

! quita

push %r3

! dejo en el stack el res

jmel %r15, 4, %r0

end: ba salida

.org 3000

arreglo: .dw 20

.end

.macro leer arg

  sll arg, 4, arg

  sra arg, 14, arg

.endmacro

(lo demás es igual, no hace falta guardar (15))

FINAL 08-10-2020

! r1:dir r2:tamanio r3:dir de lectura r4:encontrados r5:dato leído

! begin org y macros

main: add %r15, %r0, %r16 ! guardo r15 en r16

pop %r1 ! dir de inicio

pop %r2 ! tamanio

add %r1, %r0, %r3 ! dir de lectura = r1

add %r0, %r0, %r4 ! encontrados = 0

bucle: andcc %r2, %r2, %r0 ! if tamanio == 0

be fin\_bucle

ld %r3, %r5 ! lectura

call verificar

sub %r2, 1, %r2 ! tamanio--

add %r3, 4, %r3 ! dir de lectura+4

ba bucle

fin\_bucle: andcc %r4, %r4, %r0 ! if encontrados==0

be escribir

ba fin ! else termina

escribir: sethi AA2B0h, %r6

sll %r6, 2, %r6

add %r6, Bh, %r6

st %r1, %r6 ! escribe dir arreglo en AA2B000Bh

ba fin

!rutina verificadora

verificar: srl %r5, 30, %r8 ! r8:auxiliar

subcc %r8, 3, %r0 ! if r8==3

be cambiar

salir: jmpl %r15, 4, %r0

cambiar: st %r0, %r3 ! escribo 0 en dir de lectura actual

add %r14, 1, %r14

ba salir

!end rutina

fin: jmpl %r16, 4, %r0 ! termina programa