Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

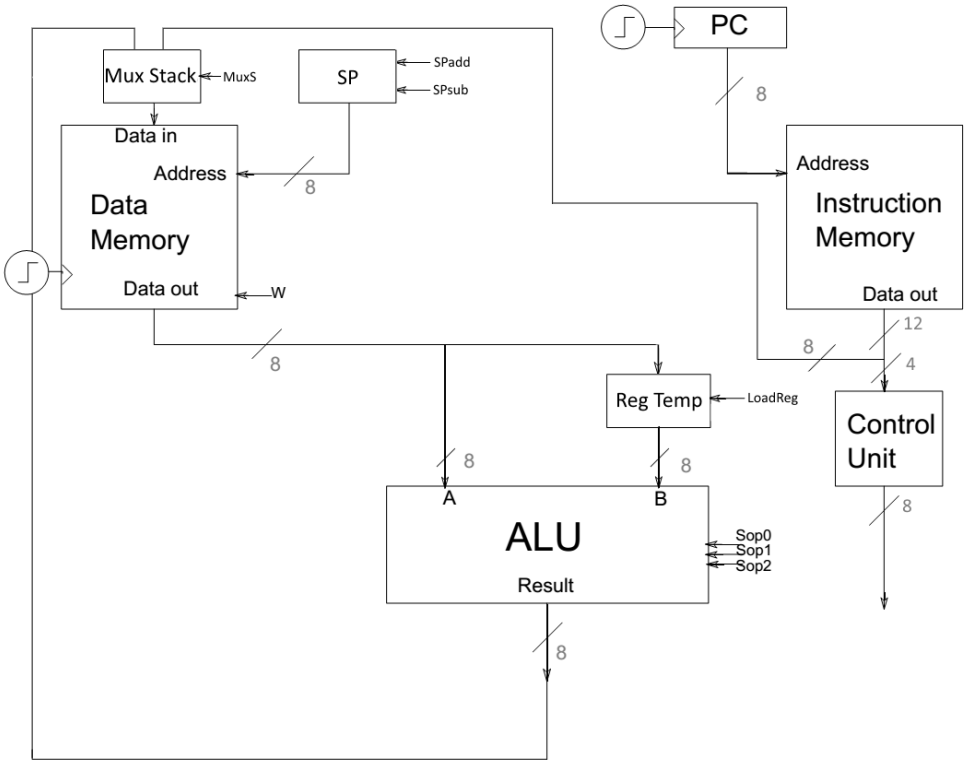
IIC2343 – Arquitectura de Computadores (II/2014)

Solución Examen

Pregunta 1

Una máquina de stack es un computador que utiliza un stack en vez de registros para almacenar los resultados de las operaciones. Esto significa que cada instrucción aritmética o lógica de dos parámetros, toma los dos valores en el tope del stack y luego los elimina, sustituyéndolos por el valor de la operación recién realizada. Para el caso de las operaciones de un parámetro, por ejemplo NOT, el computador sólo sustituye en valor en el tope del stack por el nuevo valor. Además, una máquina de stack es capaz de cargar valores literales en el tope del stack y también descartarlos.

a) Diagrame la microarquitectura de una máquina de stack de 8 bits. Este computador debe ser capaz de realizar las mismas operaciones aritméticas y lógicas que el computador básico. No es necesario tener soporte para saltos. **(4 ptos.)**
Solución:



b) Describa una ISA para la máquina del ítem anterior. Indique opcodes, señales de control y las instrucciones del assembly correspondientes. **(2 ptos.)**
Solución: En la tabla, se asume que las señales de control no indicadas se encuentran en 0. El SP siempre apunta al último elemento agregado al stack.

Instrucción	Opcode	Señales Control
PUSH Lit	0000	SPsub=1
	0001	W=1,MuxS=1
POP	0010	SPadd=1
ADD	0011	SPadd=1, LoadReg=1
	0100	Sop=ADD, W=1, MuxS=0
SUB	0101	SPadd=1, LoadReg=1
	0110	Sop=SUB, W=1, MuxS=0
AND	0111	SPadd=1, LoadReg=1
	1000	Sop=AND, W=1, MuxS=0
OR	1001	SPadd=1, LoadReg=1
	1010	Sop=OR, W=1, MuxS=0
XOR	1011	SPadd=1, LoadReg=1
	1100	Sop=XOR, W=1, MuxS=0
SHL	1101	Sop=SHL, W=1, MuxS=0
SHR	1110	Sop=SHR, W=1, MuxS=0
NOT	1111	Sop=NOT, W=1, MuxS=0

Pregunta 2

Una máquina acumuladora, es un tipo de computador en el cual se utiliza un sólo registro para almacenar los resultados de las operaciones aritméticas y lógicas. Una máquina acumuladora puede tener más registros (PC, SP, etc), pero sólo se puede usar el registro acumulador y la memoria o literales para realizar las operaciones. Para este ejercicio, asuma que tiene una máquina acumuladora de 8 bits, donde el registro acumulador es llamado A. Esta máquina posee las mismas instrucciones que el assembly x86, con la excepción que no existe la instrucción MOV, la cual es reemplazada por la instrucción para cargar datos en el registro acumulador, LOAD, y la instrucción para escribir datos en memoria, STORE. Recuerde que TODAS las operaciones aritméticas y lógicas almacenan su resultado en el registro acumulador, por lo que las instrucciones del assembly sólo reciben un parámetro.

a) Escriba en assembly de la máquina acumuladora, un programa que calcule el promedio de un conjunto de números enteros almacenados en memoria. **(2 ptos.)**
Solución:

```
        JMP    main
        suma     db 0
        res      db 0
        i        db 0
        dir      db
        n        db ?
        arreglo  db ?
    main:
        LEA arreglo
        ADD i
        STORE dir
        LOAD suma
        ADD [dir]
        STORE sum
        LOAD  i
        ADD   1
        STORE i
        CMP n
        JLT main
        LOAD suma
        DIV n
        STORE res
```

b) Escriba en assembly de la máquina acumuladora, un programa que calcule un histograma de un conjunto de números enteros no negativos, almacenados en memoria. Asuma que los números se encuentran el intervalo $[0, 255]$ y que el número de casilleros del histograma siempre será un divisor de 256. Cualquier detalle de implementación o aspecto que se asuma debe quedar claramente explicado. **(4 ptos.)**

Solución:

```
        JMP   main
        i           db 0
        bin         db
        dir         db
        casilleros  db ?
        ancho       db
        n           db ?
        arreglo     db ?
        .
        .
        .
        res         db 0
        .
        .
        .
    main:
        LOAD 256
        DIV  casilleros
        STORE ancho
    loop:
        LEA arreglo
        ADD i
        LOAD [A]
        DIV ancho
        STORE bin

        LEA res
        ADD bin
        STORE dir
        LOAD [A]
        ADD 1
        STORE [dir]

        LOAD  i
        ADD   1
        STORE i
        CMP n
        JLT loop
```

Pregunta 3

a) Describa la relación entre direcciones virtuales y físicas, páginas y marcos. Describa además el proceso que relaciona a todos estos elementos, nombrando todas las piezas de hardware que participan. **(1 pto.)**
Solución: La direcciones virtuales son posiciones de memoria que no existen fisicamente y que pueden estar asociadas a una dirección de memoria física, si este mapeo existe en la tabla de páginas. El espacio direccionable de la memoria virtual es igual al especio direccionable del computador. Las páginas son espacios contiguos de memoria virtual (direcciones virtuales), que están asociadas a un marco físico (direcciones físicas), mediante un mapeo dado por una tabla de páginas. La tranformación es realizada por una unidad ubicada en la CPU llamada MMU, que lee el contenido de su caché (TLB) o desde la tabla de páginas.

b) Un computador utiliza palabras de 8 bits, tiene un espacio direccionable de 64GB y una memoria principal de 2GB dividida en marcos de 2KB. ¿Cuántos bits se necesitan para describir el número de página virtual y el número de marco físico? Si todos los flags de la tabla de páginas toman 12 bits por entrada, ¿cuantos bits de espacio utiliza una tabla de páginas? **(1 pto.)**
Solución: Se necesitan 25 bits para describir el número de página y 20 bits para el número de marco físico. La tabla de páginas utiliza $2^25 \times (12 + 20) = 2^25 \times 2^5 = 2^30$ bits.

c) Complete la siguiente tabla asumiendo que por cada entrada de la tabla de páginas, se utilizan 4 bits para flags: **(1 pto.)**
Solución:

Bits Dir. Virt.	Bits Dir. Fís.	Tam. Pág.	Bits Pág.	Bits Marco	Bits por entrada
32	32	16KB	18	18	22
32	26	8KB	19	13	17
36	32	32KB	21	17	21
40	36	32KB	25	21	25
64	40	64KB	48	24	28

d) ¿Qués es un cambio de contexto y un PCB? ¿Que relación existe entre ellos? **(1 pto.)**
Solución: Un cambio de contexto es el proceso que realiza el sistema operativo para cambiar la ejecución de un proceso a otro. El PCB (Process Control Block) es una estructura almacenada por el sistema operativo, donde cada proceso tiene una y contiene todos los datos relevantes que definen su estado. El sistema operativo utiliza el PCB para respaldarrestaurar los procesos cuando hay un cambio de contexto y para tomar decisiones relacionadas con scheduling.

e) Usando la siguiente tabla, calcule los momentos en que se ejecuta y termina cada proceso, usando los esquemas FiFo y Round Robin con un intervalo de ejecución de 60. Además, calcule para cada esquema el tiempo total de espera por atención. ¿Cuál de los dos esquemas es más eficiente con respecto a esta métrica? **(2 ptos.)**

Proceso	P1	P2	P3	P4	P5
Tpo. Ejecución	120	60	180	50	300

Solución:
Fifo:

- P1: Inicio=0, Fin=120 – Espera=0
- P2: Inicio=120, Fin=180 – Espera=120
- P3: Inicio=180, Fin=360 – Espera=180
- P4: Inicio=360, Fin=410 – Espera=360
- P5: Inicio=410, Fin=710 – Espera=410

Espera total = 1070

Round Robin:

- P1: Inicio=0, Fin=60 – Inicio=290, Fin=350 Espera=230
- P2: Inicio=60, Fin=120 – Espera=60
- P3: Inicio=120, Fin=180 – Inicio=350, Fin=410 – Inicio=470, Fin=530 – Espera=350
- P4: Inicio=180, Fin=230 – Espera=180
- P5: Inicio=230, Fin=290 – Inicio=410, Fin=470 – Inicio=530,Fin=590 – Inicio=590, Fin=650 – Inicio=650, Fin=710 – Espera=410

Espera total = 1230
Luego, para este caso, es más eficiente FiFo.

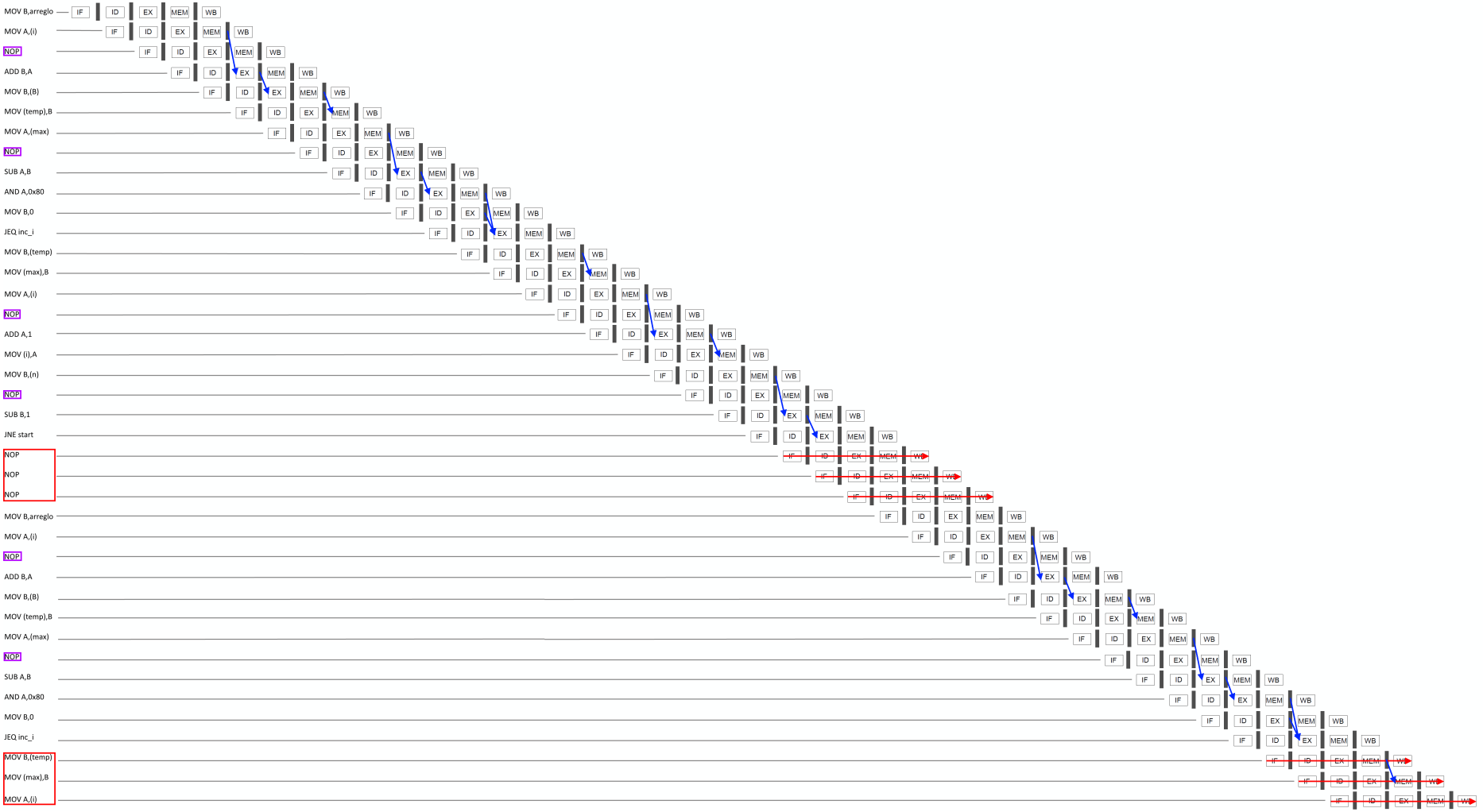
Pregunta 4

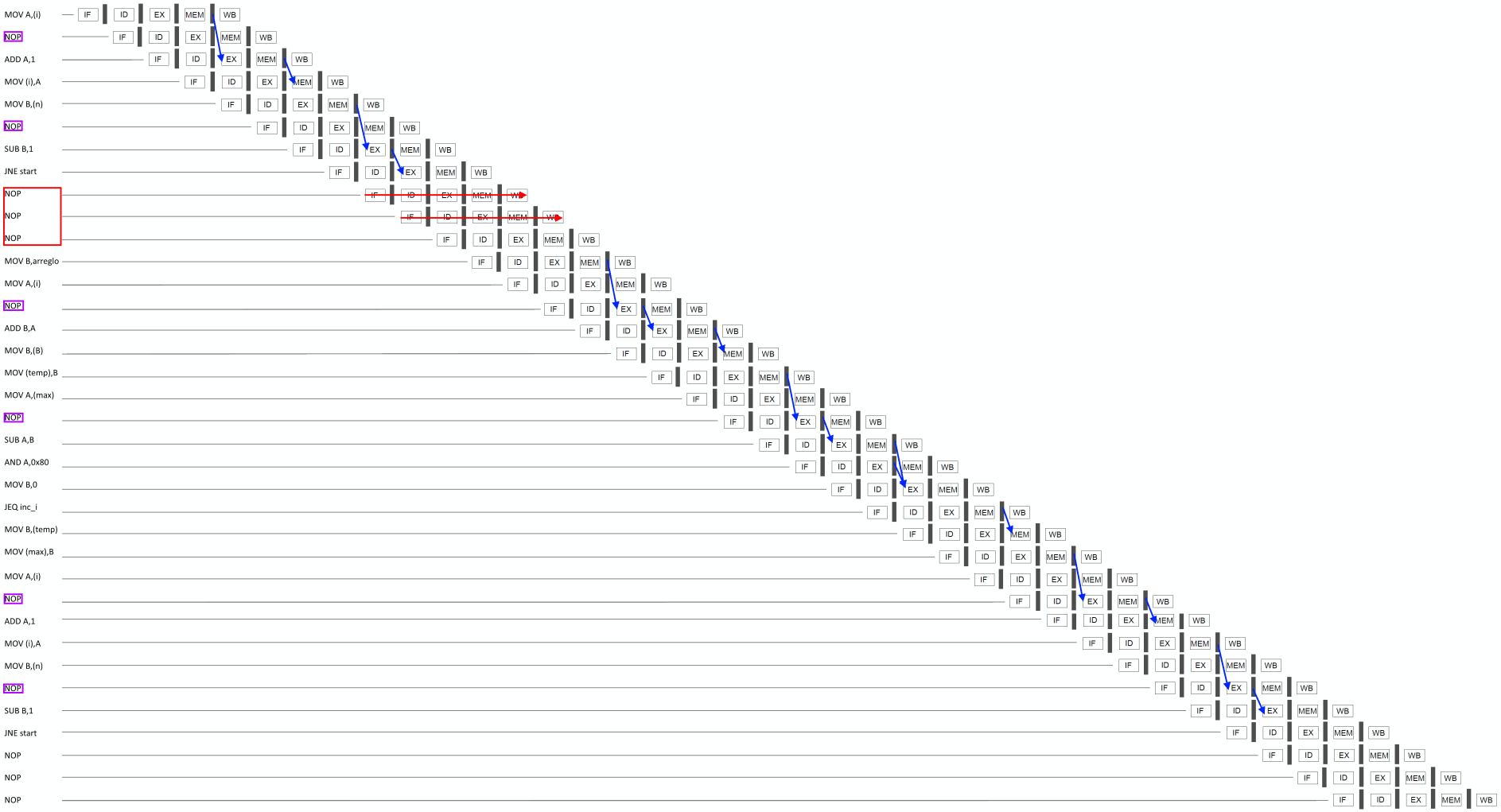
a) Describa los tipos de paralelismo SIMD y SISD. ¿Qué tipo de operaciones son las que más beneficio sacan del paralelismo SIMD?¿Es posible tener simultáneamente paralelismo del tipo SISD y SIMD? **(1 pto.)**
Solución: El paralelismo SIMD (Single Instruction Multiple Data) ocurre cuando la misma instrucción/-programa debe ser aplicado a datos distintos. Las operaciones vectoriales y matriciales son las que mayor provecho sacan de esto.
SISD ocurre cuando se realiza la ejecución paralela de un programa sobre un mismo dato. Generalmente este se logra mediante pipelining. Operaciones largas que pueden ser divididas en sub-operaciones son las que mayor ventaja obtienen de SISD.
Para que ambos esquemas, SISD y SIMD, puedan ejecutarse simultaneamente, basta que las unidades de ejecución del pipeline, sean capaces de aplicar la misma operación a múltiples datos distintos.

b) Escriba un programa para el computador básico con pipeline, que encuentre el máximo valor en un arreglo de n números enteros. **(2 ptos.)**
Solución:

```
DATA
    i         0
    max       0
    temp
    n         ?
    arreglo   ?
CODE
    start:
              MOV B, arreglo
              MOV A, (i)
              ADD B, A
              MOV B, (B)
              MOV (temp), B
              MOV A, (max)
              SUB A, B
              AND A,0x80
              MOV B, 0
              JEQ inc_i
              MOV B, (temp)
              MOV (max), B
    inc_i:
              MOV A, (i)
              ADD A, 1
              MOV (i), A
              MOV B, (n)
              SUB B, 1
              JNE start
```

c) Determine el número de ciclos que se demora el código anterior, usando el arreglo [3, 1, 7]. Detalle en un diagrama los estados del pipeline por instrucción. El pipeline tiene forwarding entre todas sus etapas, el manejo de stalling es por software (instrucción NOP) y predicción de salto asumiendo que no ocurre. Indique en el diagrama cuando ocurre forwarding, stalling y flushing. ¿Cuántos ciclos tomaría el programa si se agrega el número 2 al final del arreglo?**(3 ptos.)**
Solución: En base al siguiente diagrama, el programa toma 80 ciclos. Si se agrega un 2 al final del arreglo, el comportamiento es el mismo de la segunda iteración, la cual toma 26 ciclos. Luego el programa tomaría 106 ciclos.


MOV B,arreglo
MOV A,(i)
NOP
ADD B,A
MOV B,(B)
MOV (temp),B
MOV A,(max)
NOP
SUB A,B
AND A,0x80
MOV B,0
JEQ inc_i
MOV B,(temp)
MOV (max),B
MOV A,(i)
NOP
ADD A,1
MOV (i),A
MOV B,(n)
NOP
SUB B,1
JNE start
NOP
NOP
NOP
MOV B,arreglo
MOV A,(i)
NOP
ADD B,A
MOV B,(B)
MOV (temp),B
MOV A,(max)
NOP
SUB A,B
AND A,0x80
MOV B,0
JEQ inc_i
MOV B,(temp)
MOV (max),B
MOV A,(i)
IF
ID
EX
MEM
WB


MOV A,(i)
NOP
ADD A,1
MOV (i),A
MOV B,(n)
NOP
SUB B,1
JNE start
NOP
NOP
NOP
MOV B,arreglo
MOV A,(i)
NOP
ADD B,A
MOV B,(B)
MOV (temp),B
MOV A,(max)
NOP
SUB A,B
AND A,0x80
MOV B,0
JEQ inc_i
MOV B,(temp)
MOV (max),B
MOV A,(i)
NOP
ADD A,1
MOV (i),A
MOV B,(n)
NOP
SUB B,1
JNE start
NOP
NOP
NOP
IF
ID
EX
MEM
WB