DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2019)

Pauta Interrogación 1

Respuestás sin desarollo o justificación no tendrán puntaje.

Jueves 26 de Septiembre a las 18:30 horas

Forma A|H|L|I

Instrucciones

Lea atentamente los enunciados. Responda cada preguntas en hojas separadas. Ponga su nombre, numero de alumno y forma de la prueba a cada una de las hoja de respuesta. Siga el código de honor.

Pregunta 1 (1,5 ptos.)

1. **(0,2 ptos.)** ¿Cuál es el paradigma arquitectónico de la microarquitectura del computador básico según lo visto en clases? ¿Por qué? De un ejemplo real del otro paradigma arquitectónico visto en clases.

   Respuesta: Corresponde al paradigma arquitectónico Harvard. Este se caracteriza por tener una memoria dedicada a las instrucciones (ROM) y otra a los datos (RAM). Esto permite tener tamaños distintos para las palabras de datos e instrucciones, entre otras ventajas.
   (0,1 Punto)

   Otra arquitectura es la Von Neumann, que tiene la misma memoria para datos e instrucciones. Esto permite escribir tus propios programas. Los procesadores de Intel utilizan este último paradigma (el primero de ellos en usarlo fue el Intel 80186). (0,1 Punto)

2. **(0,2 ptos.)**¿Porqué es conveniente utilizar la representación hexadecimal al definir números binarios?

   Respuesta: Esto se debe a la facilidad para pasar un número de una base a otra: De binario a hexadecimal basta agrupar bits de a 4. Cada uno de estos grupos corresponde a un dígito en hexadecimal. Para pasar de hexadecimal a binario es el proceso inverso: Cada dígito en hexadecimal se pasa a un número binario de 4 bits, y luego se juntan para obtener el resultado. Otra clara ventaja es que ocupan menos espacio visual (en un bit hexadecimal podemos representar 4 bits en binario) (0,2 Puntos por cualquiera de las dos razones)

3. **(0,4 ptos.)** Convierta a decimal el número de 8 bits 0x?? interpretándolo como un número binario sin signo, con signo, en complemento a 1 y en complemento a 2.

Respuesta:

Procedimientos Comunes a todas las formas (A, I, H, L)

- **binario sin signo:** Multiplicamos por potencias de 2 según la posición de los 1s presentes en el número, en donde el bit menos significativo corresponde a 2^0 y el bit n corresponde a 2^{n-1}
  (0,1 Puntos por el desarrollo y resultado correcto)
- **binario con signo:** El bit más significativo sólo nos indica el signo. Si es un 1, corresponde a un negativo, y viceversa. Por lo tanto, lo calculamos como un binario sin signo (quitando el último bit), y luego agregamos el signo según corresponda.
  (0,1 Puntos por el desarrollo y resultado correcto)
- **complemento a 1:** el bit más significativo indica sólo el signo. Si es un 1, corresponde a un negativo, y viceversa. En caso de ser negativo, para obtener el número positivo equivalente, cambiamos los 1s por 0s, y viceversa. Luego calculamos como si fuera un binario de 7 bits (no incluimos el bit de signo), y tomamos el inverso aditivo.
  (0,1 Puntos por el desarrollo y resultado correcto)
- **complemento a 2:** Nuevamente el bit más significativo nos indica el signo. Si el número es negativo, para obtener el binario positivo de igual magnitud, recorremos el número desde el bit menos significativo hasta encontrar el primer 1. Este lo dejamos intacto, y el resto de los bits los cambiamos (1s por 0s y viceversa).[1] Luego el cálculo es el acostumbrado para números binarios, agregando el signo según corresponda.
  (0,1 Puntos por el desarrollo y resultado correcto)

Forma A

0x8F corresponde a: 10001111

- **binario sin signo:**
  $= 1 \times 2^7 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  $= 128 + 8 + 4 + 2 + 1 =$ **143**
- **binario con signo:**
  -$(1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = -(8 + 4 + 2 + 1) =$ **-15**
- **complemento a 1:**
  $10001111 \rightarrow 1110000 \rightarrow -(1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4) = -(64 + 32 + 16) =$ **-112**
- **complemento a 2:**
  $01110001 \rightarrow -(1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^0) = -(64 + 32 + 16 + 1) =$ **-113**

Forma H

0x91 corresponde a: 10010001

- **binario sin signo:**
  $1 \times 2^7 + 1 \times 2^4 + 1 \times 2^0 = 128 + 16 + 1 =$ **145**

[1] Esto es equivalente a cambiar todos los bits, y luego sumar 1

- **binario con signo:**
  -$(1 \times 2^4 + 1 \times 2^0 = -(16 + 1) =$ **-17**
- **complemento a 1:**
  $10010001 \rightarrow 1101110 \rightarrow -(1\times2^6+1\times2^5+1\times2^3+1\times2^2+1\times2^1) = -(64+32+8+4+2) =$ **-110**
- **complemento a 2:**
  $01110001 \rightarrow -(1\times2^6+1\times2^5+1\times2^3+1\times2^2+1\times2^1+1\times2^0) = -(64+32+8+4+2+1) =$ **-111**

Forma I

0x90 corresponde a: 10010000

- **binario sin signo:**
  $1 \times 2^7 + 1 \times 2^4 = 128 + 16 =$ **144**
- **binario con signo:**
  -$(1 \times 2^4) =$ **-16**
- **complemento a 1:**
  $10010000 \rightarrow 1101111 \rightarrow -(1\times2^6+1\times2^5+1\times2^3+1\times2^2+1\times2^1+1\times2^0) = -(64+32+8+4+2+1) =$ **-111**
- **complemento a 2:**
  $01110000 \rightarrow -(1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4) = -(64 + 32 + 16) =$ **-112**

Forma L

0x92 corresponde a: 10010010

- **binario sin signo:**
  $1 \times 2^7 + 1 \times 2^4 + 1 \times 2^1 = 128 + 16 + 2 =$ **146**
- **binario con signo:**
  -$(1 \times 2^4 + 1 \times 2^1) = -(16 + 2) =$ **-18**
- **complemento a 1:**
  $10010010 \rightarrow 1101101 \rightarrow -(1\times2^6+1\times2^5+1\times2^3+1\times2^2+1\times2^0) = -(64+32+8+4+1) =$ **-109**
- **complemento a 2:**
  $01101110 \rightarrow -(1 \times 2^6 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1) = -(64 + 32 + 8 + 4 + 2) =$ **-110**

4. **(0,4 ptos.)** Convierta a decimal el número de 32 bits 0x???00000 como un float de 32 bits del estandar IEEE754.

Respuesta:

Procedimientos Comunes a todas las formas (A, I, H, L)

El bit más significativo corresponde al signo del significante, los 8 bits siguientes corresponden al exponente. Este está desplazado en 127, lo que quiere decir que al obtener el exponente en decimal a partir del número de 8 bits, debemos restar 127 para obtener el exponente correspondiente [2]. Luego los 23 bits menos significativos corresponden al significante, el cual se guarda normalizado, y sin el 1 a la izquierda de la coma.
(0,4 Puntos por desarrollo y resultado correcto.)

Por lo tanto el resultado sería:

$(-1)^{signo} \times 1, \text{significante} \times 2^{(\text{exponente}-127)}$

Donde el signo es $-$ si el bit correspondiente es 1 y viceversa.

Forma A

0x3F800000 corresponde a: 00111111100000000000000000000000

- **bit de signo:** 0, es decir es positivo
- **exponente:** $01111111 \rightarrow 127 - 127 = 0$
- **significante:** $00000000000000000000000 \rightarrow 0$

Por lo tanto el resultado es: $1,0 \times 2^0 = 1$

Forma H

0xBF800000 corresponde a: 10111111100000000000000000000000

- **bit de signo:** 1, es decir es negativo
- **exponente:** $01111111 \rightarrow 127 - 127 = 0$
- **significante:** $00000000000000000000000 \rightarrow 0$

Por lo tanto el resultado es: $-1,0 \times 2^0 = -1$

Forma I

0x40800000 corresponde a: 01000000100000000000000000000000

- **bit de signo:** 0, es decir es positivo
- **exponente:** $10000001 \rightarrow 129 - 127 = 2$
- **significante:** $00000000000000000000000 \rightarrow 0$

Por lo tanto el resultado es: $1,0 \times 2^2 = 4$

[2] es una manera de tener números negativos sin perder el bit de signo, ni usar complemento a 2

Forma L

0xC0800000 corresponde a: 11000000100000000000000000000000

- **bit de signo:** 1, es decir es negativo
- **exponente:** $10000001 \rightarrow 129 - 127 = 2$
- **significante:** $00000000000000000000000 \rightarrow 0$

Por lo tanto el resultado es: $-1,0 \times 2^2 = -4$

5. **(0,3 ptos.)** Al leer 2 palabras contiguas de 1 byte de una memoria en orden Big Endian, se obtiene el número ???. ¿Qué número se obtendría al leer estas palabras en orden Little Endian?

Respuesta:

Procedimientos Comunes a todas las formas (A, I, H, L)

El formato Big Endian contempla los bytes del más significativo al menos significativo, y el formato Little Endian lo hace al revés. Para obtener el número pedido podemos calcular ambas palabras pasando de base 10 a base 2, luego dividir estos 16 bits en dos bytes, darlas vuelta, y luego pasar este número a base 10 nuevamente:
(0,3 Puntos por desarrollo y resultado correcto.)

Forma A

- **Big Endian:** 260 → 0000000100000100 → 00000001|00000100
- **Little Endian:** 00000100|00000001 → 0000010000000001 → 1025

Forma H y L

- **Big Endian:** 258 → 0000000100000010 → 00000001|00000010
- **Little Endian:** 00000010|00000001 → 0000001000000001 → 513

Forma I

- **Big Endian:** 513 → 0000001000000001 → 00000010|00000001
- **Little Endian:** 00000001|00000010 → 0000000100000010 → 258

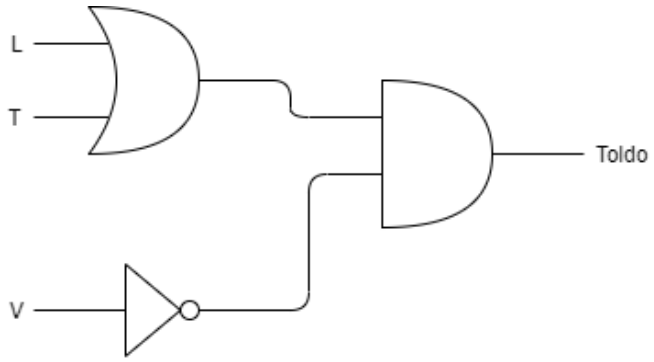
Pregunta 2 (1.0 ptos.)[3]

Su profesor, debido al calor venidero, busca construirse un toldo que se active o desactive según tres sensores, independientes, de viento, temperatura y luz. El objetivo es que funcione de la siguiente forma:

- El toldo se activará si hay luz y temperatura, pero no viento.
- El toldo se activará si hay luz, pero no temperatura ni viento.
- El toldo se activará si no hay luz, hay temperatura, pero no existe viento.

Construya una tabla de verdad con la información proporcionada (0.3 ptos) y, con la misma, diseñe un circuito con las compuertas lógicas básicas, vistas en clases, que resuelvan el problema. (0.7 ptos)

Respuesta:



V	T	L	Toldo
0	1	1	1
0	0	1	1
0	1	0	1
X	X	X	0

Tabla 1: Tabla de verdad

$$\bar{V}TL + \bar{V}\bar{T}L + \bar{V}T\bar{L} = Toldo \tag{1}$$

[3]No es 1.2 ptos., es solo 1.0 pto.

Pregunta 3 (1.5 ptos.)

El siguiente programa escrito en C implementa un algoritmo de conteo de los bits distintos entre dos números. Traduzca el programa al assembly del computador básico, cumpliendo lo siguiente:

- Su programa debe ser una traducción directa del programa en C, es decir, además de traducir la funcionalidad (conteo de los bits distintos) debe hacerlo implementando el mismo algoritmo descrito en el programa, incluyendo las funciones como subrutinas.
- Su programa debe definir (al menos) las variables a, b que se ocuparán como operandos para el conteo de los bits distintos, y la variable c que almacená el resultado.

```
1  unsigned char contarBitsSeteados(unsigned char n)
2  {
3      unsigned char cuenta = 0;
4      while (n != 0)
5      {
6          cuenta += n & 1;
7          n >>= 1;
8      }
9      return cuenta;
10 }
11
12 unsigned char contarBitsDistintos(unsigned char a, unsigned char b)
13 {
14     return contarBitsSeteados(a^b);
15 }
16
17 void main()
18 {
19     unsigned char a = 10;
20     unsigned char b = 20;
21     unsigned char c = 0;
22     c =  contarBitsDistintos(a, b);
23 }
```

Respuesta:

```
DATA:  // (0.1 ptos.)
  a      10
  b      20
  c      0
  cuenta 0

CODE:  // (0.2 ptos.)
  MOV A, (a)
  MOV B, (b)
  CALL contarBitsDistintos
  MOV (c), A
  JMP end

contarBitsDistintos: // (0.2 ptos.)
  XOR A, B
  CALL contarBitsSeteados
  RET

contarBitsSeteados: // (0.8 ptos.)
  CMP A, 0
  JEQ noWhile
  MOV B, A
  AND A, 1
  ADD A, (cuenta)
  MOV (cuenta), A
  MOV A, B
  SHR A,A
  JMP contarBitsSeteados
noWhile:
  MOV A, (cuenta)
  RET

end:
 JMP end
```

Se espera que exista una correspondencia entre los códigos, algunos de los descuentos aplicados por faltar a esto son:

- 0.1 No definir la sección de DATA.
- 0.1 Mover puntero envés del valor.
- 0.2 No llamar a la función principal.
- 0.4 Por cada función no implementada como subrutina.
- 0.3 Equivocarse en el tipo de salto.
- 0.2 Implementación incorrecta o falta de CMP.
- 0.2 No inicializar variables.
- 0.3 Hacer un DoWhile en vez de While.
- 0.1 Escribir CALL en lugar de JMPs.
- 0.2 No usar XOR.
- 0.2 No guardar el resultado en variable C.
- 0.1 Si ejecutan el código de las subrutinas sin llamarlas.

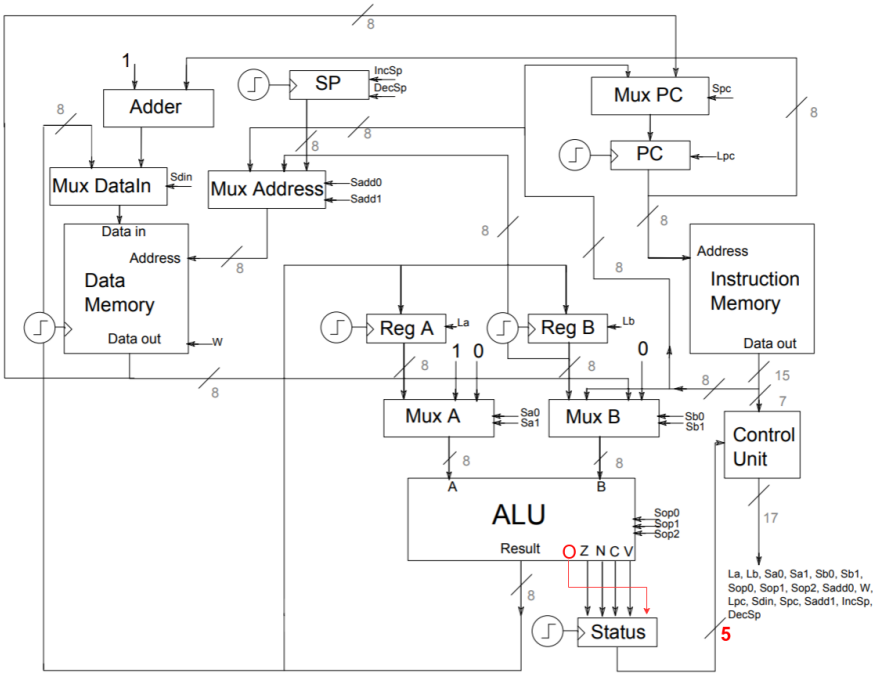
Pregunta 4 (2 ptos.)

Se requiere agregar las siguientes instrucciones al computador básico. Para cada caso indique si es posible agregarlas sin modificar el hardware del computador, en cuyo caso debe especificar las señales de control que se deben habilitar para ejecutar la instrucción. En caso de que se requiera modificar el computador, realice los cambios en el diagrama adjunto, especifique las nuevas señales de control agregadas (si corresponde) e indique que señales de control se deben habilitar para ejecutar la instrucción.

1. **(0,5 ptos.)** Agregue la instrucción `JPO Dir` la cual salta a la instrucción asociada al label `Dir` si el resultado de la instrucción anterior fue impar.

   `RESPUESTA:`

   - No es posible agregar esta instrucción sin modificar el hardware del computador. Esto debido a que los saltos condicionales existentes en el computador básico ocupan las señales: `Z` (resultado de la ALU es cero), `N` (el bit más significativo del resultado de la ALU, indicando que este es negativo cuando es uno), `C` (pérdida de un bit) y `V` (overflow). Ninguna de esta señales, ni mezcla de ellas nos indica que el resultado es impar, por lo que es necesario agregar hardware. **(+ 0,1 pts.)**
   - Modificación: **(+ 0,2 pts.)**



   - Especificación de señales agregadas: **(+ 0,1 pts.)**

     Se agrega la señal condicional `O`, la cuál será el bit menos significativo del resultado de la ALU, indicado que es impar cuando es uno.
   - Indicación de señales de control a ejecutar: **(+ 0,1 pts.)**

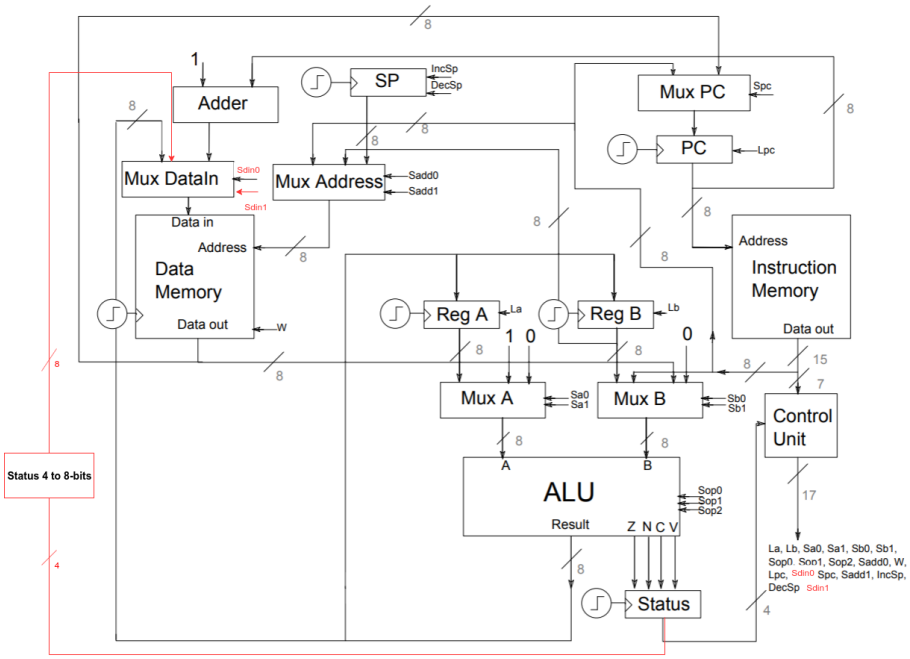
     NOTA: Toda señal de cargado que no esta en la tabla (como La, Lb, W, etc) son cero.

Instrucción	Condición	Lpc	Spc0
JPO	O=1	1	LIT

2. **(0,5 ptos.)** Agregue la instrucción `PUSHF` la cual almacena el valor del registro Status en el Stack.

   `RESPUESTA:`

   - No es posible agregar esta instrucción sin modificar el hardware del computador. Esto debido a que el valor del registro Status solo es recibida por la Control Unit, sin tener otra comunicación con algún otro componente para poder entregarle su valor a la memoria de datos. **(+ 0,1 pts.)**
   - Modificación: **(+ 0,2 pts.)**



   NOTA: El componente Status 4 to 8-bits, lo único que hace es agregar cuatro bits en cero al valor entregado por Status, para poder ingresarlo al selector sin problemas.

   - Especificación de señales agregadas: **(+ 0,1 pts.)**

     Se agrega la señal condicional `Sdin1`, la cuál será un bit más para poder seleccionar más de dos opciones en el Mux Datain.

     NOTA: En caso de realizar la modificación que conecta al Mux A, la sección de indicaciones de señales de control vale el doble, pues no corresponde indicar que hay que agregar nuevas señales.

   - Indicación de señales de control a ejecutar: **(+ 0,1 pts.)**

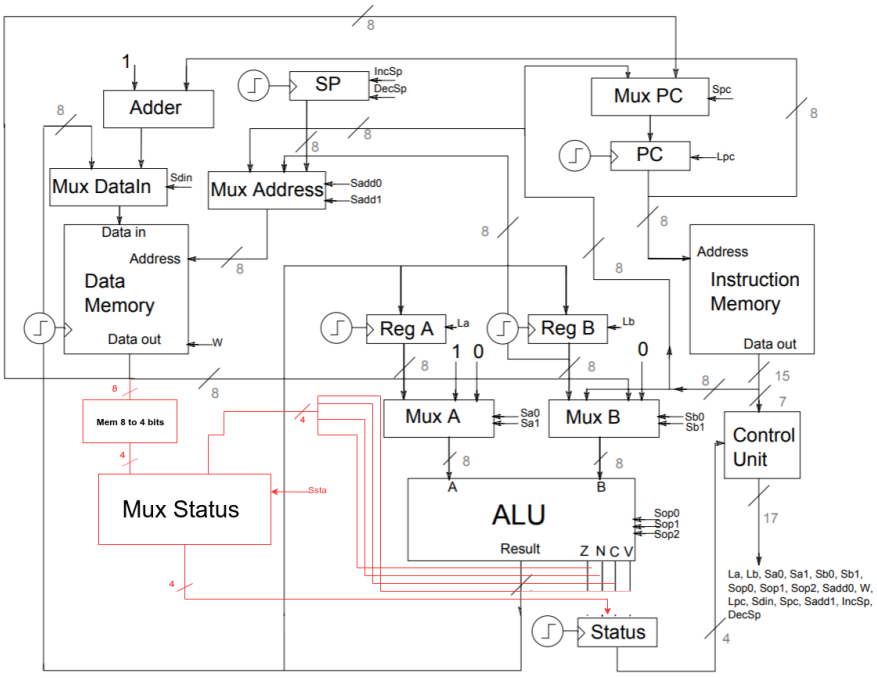
     NOTA: Toda señal de cargado que no esta en la tabla (como La, Lb, Lpc, etc) son cero.

Instrucción	**Sdin0,1**	**Sadd0,1**	**W**	**DecSp**
`PUSHF`	`Status`	`SP`	`1`	`1`

3. **(0,5 ptos.)** Agregue la instrucción `POPF` la cual guarda en el registro Status el último valor almacenado en el Stack.

   `RESPUESTA:`

   - No es posible agregar esta instrucción sin modificar el hardware del computador. Esto debido a que el valor del registro Status solo recibe señales condicionales de operaciones de la ALU, no tiene acceso a la salida de memoria de datos. **(+ 0,1 pts.)**
   - Modificación: **(+ 0,2 pts.)**



   NOTA: El componente Mem 8 to 4-bits, lo único que hace es entregar los bits menos significativos de su entrada de 8 bits.

   - Especificación de señales agregadas: **(+ 0,1 pts.)**
     Se agrega la señal condicional `Ssta`, la cuál será un bit más para poder seleccionar los valores de entrada al registro Status.
   - Indicación de señales de control a ejecutar: **(+ 0,1 pts.)**
     NOTA: Toda señal de cargado que no esta en la tabla (como La, Lb, Lpc, etc) son cero.

Instrucción	IncSp	Ssta	Sadd0,1
POPF	1	-	-
	0	Dout	SP

4. **(0,5 ptos.)** Agregue la instrucción `MOV B,((Dir))` la cual guarda en el registro B el valor en memoria en la posición indicada por la posición memoria asociada al label `Dir`. (`B = MEM[MEM[Lit]]`)

   `RESPUESTA:`

   - Si es posible agregar esta instrucción sin modificar el hardware del computador, en especifico esta instrucción se puede realizar ejecutando dos ciclos. Primero ejecutando las mismas señales de la instrucción `MOV B, (Dir)`, y finalmente en el siguiente ciclo ejecuta las mismas señales de la instrucción `MOV B, (B)`. **(+ 0,2 pts.)**
   - Indicación de señales de control a ejecutar: **(+ 0,3 pts.)**

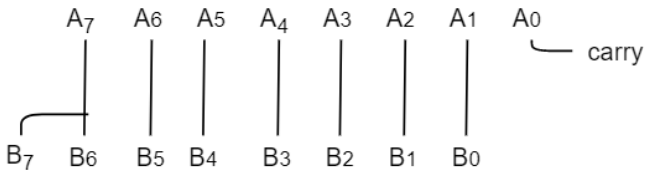
     NOTA: Toda señal de cargado que no esta en la tabla (como La, Lpc, W, etc) son cero.

Instrucción	**Lb**	**Sadd0,1**	**Sop0,1,2**	**Sb0,1**	**Sa0,1**
MOV	1	LIT	ADD	Dout	ZERO
B,((Dir))	1	B	ADD	Dout	ZERO

Pregunta Bonus.1 (0,7 ptos.)

En algunas arquitecturas existen dos operaciones para hacer shifting de números en complemento a 2, una de ellas es `SAR`, que corresponde a un shift right donde se preserva el signo del número.

1. **(0,5 ptos.)** Diseñe un circuito que realice la operación SAR sobre 8 bits.



2. **(0,2 ptos.)** Agréguela a la ISA del computador básico con los mismos tres casos para `SHR`, indicando las señales de control.

   Básicamente es agregar una cuarta señal de control en la ALU. Pudiendo ser 1000, 1001, 1010 algunas señales propuestas dejando el resto de la ALU como el estándar.

Pregunta Bonus.2 (0,3 ptos.)

1. **(0,1 ptos.)** En el sistema de numeración que usamos, con base diez, ¿por qué después del 9 viene el 10?

   El sistema de numeración en base diez consta de los símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

   0 ->cero
   1 ->uno
   ...
   9 ->nueve

   Las cantidades se representan como una concatenación de estos símbolos, que llamamos dígitos, de largo variable y ordenados según el dígito más o menos significativo. Un mismo símbolo, según la posición en la que esté, tomará su valor final.

   Cada posición vale diez veces más que la posición que tiene la cifra inmediatamente menos significativa y, al final, la cantidad que se representa es la suma del valor de todos los dígitos.

   EJEMPLO: [No obligatorio, pero sirve para la explicación]
   Tomemos el número XYZ, donde Z es su dígito menos significativo. Como no hay una posición menos significativa que Z, Z vale Z. Para Y existe una posición menos significativa, por lo que Y vale diez por Y. Para X existen dos posiciones menos significativas, por lo que X vale diez por diez por X. Y al final, el valor del número es Z + diez * Y + diez * diez * X.

   Ahora, queremos representar la cantidad 9 + 1, el número que viene después del nueve, pero no hay una cifra que valga esa cantidad. Sin embargo, si miramos nuestro sistema, podemos decir que:

diez = XY = diez * X + Y

Si X es 1 e Y es 0, la única forma de representar la cantidad "diez.[en] este sistema, vemos que diez se escribe como "10".

Es por eso que después del 9 viene el 10.

2. Acotando los números a un máximo de cinco dígitos, transforme los siguientes números decimales a su representación con complemento a 10.
   - **(0,1 ptos.)** 195 = 195
   - **(0,1 ptos.)** -4320 = 95680