UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
4 de octubre de 2019 **Grupo 81 Ex 1 A**

Ejercicio 1. Considere un computador de 64 bits que direcciona la memoria por bytes, que dispone de 32 registros y un juego de 136 instrucciones máquina. Responda a las siguientes preguntas:

a) ¿Cuál es el rango de números enteros en complemento a 2 que puede manejar?
b) Exprese en MB el espacio de direcciones de este computador.
c) ¿Cuál es el tamaño en bits de cada registro?
d) ¿Cuántos bits se necesitan para identificar a un registro en este computador?
e) Dada una instrucción máquina, indique el número de bits necesarios para representar el código de operación.

Solución:

a) En un computador de 64 bits los enteros ocupan 64 bits. El rango de número enteros en complemento a 2 es de $[-2^{63}, 2^{63}-1]$.
b) En un computador de 64 bits las direcciones de memoria ocupan 64 bits y el número de posiciones de memoria que se pueden direccionar es 2^{64}. El número de bytes, por tanto, que se pueden direccionar es $2^{64} = 2^{64} / 2^{20} = 2^{34}$ MB.
c) En un computador de 64 bits los registros ocupan 64 bits.
d) Como el computador dispone de 32 registros se necesitan $\log_2 32 = 5$ bits.
e) Como el computador dispone de 136 instrucciones máquina, se necesitan 8 bits para poder codificarlas. Con 7 bits se podrían representar $2^7 = 128$ instrucciones solo.

Ejercicio 2. Represente en el estándar IEEE de **doble** precisión el número -18,25. Exprese el resultado en hexadecimal.

Solución:

$-18{,}25_{(10} = -10010{,}01_{(2} = -1{,}001001 \times 2^4$ (de forma normalizada). El bit de singo es 1 (el número es negativo). El valor del exponente es 4 + 1023 (sesgo) = 1027 = 10000000011. La mantisa que se almacena (no se almacena el bit implícito) es 001001000…..0000.

El número es 1100 0000 0011 0010 0100 00…..00000000 = 0xC032400000000000

Ejercicio 3. Dada la siguiente definición del segmento de datos en un fragmento de programa que usa el ensamblador del MIPS 32:

```
.data:
      .align 2
      N:  .word 80
      V1: .space 200
```

Responda:

a) ¿Para qué se utiliza `align`?
b) ¿Cuántos bits almacena el dato que se encuentra en la posición de memoria `N`?
c) Si `V1` es un vector de números enteros, escriba un fragmento de programa que permita imprimir el contenido de todos elementos del vector (`v1[0]`, `v1[1]` ….), cada elemento en una línea.

Solución:

a) La directiva `align` se utiliza para forzar a que la siguiente dirección de memoria que se especifique se alinee, en este caso a múltiplo de $2^2 = 4$.
b) La dirección de memoria etiquetada como `N` almacena una palabra que en el caso del MIPS 32 ocupa 32 bits.
c) Teniendo en cuenta que el vector V1 ocupa 200/4 =50 elementos, un posible fragmento de código es el siguiente:

```
          li   $t0, 0
          li   $t1, 0
          li   $t2, 50
bucle:    bge  $t1, $t2, fin
          lw   $a0, V1($t1)
          li   $v0, 1
          syscall        # se imprime el elemento
          li   $a0, '\n'
          li   $v0, 11
          syscall        # se imprime el salto de línea.

          addi $t0, $t0, 1
          addi $t1, $t1, 4
          b bucle
fin:
```

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
4 de octubre de 2019 **Grupo 81 Ex 1 B**

Ejercicio 1. Considere un hipotético computador con 100 registros que direcciona la memoria por bytes. En este computador se pueden direccionar como mucho 64 KB de memoria. Asumiendo que el tamaño de la palabra de este computador coincide con el número de bits empleados para las direcciones de memoria. Responda a las siguientes preguntas:

a) ¿Cuántos bits se emplean para las direcciones de memoria?
b) ¿Cuál es el tamaño de los registros?
c) ¿Cuántos bits se almacenan en cada posición de memoria?
d) Si el computador fuera de 32 bits. ¿Cuál sería el tamaño de la máxima memoria direccionable? Exprese el resultado en MB.
e) ¿Cuántos bits se necesitan para identificar a los registros?

Solución:

a) Para direccionar como mucho 64KB = 2^{16} bytes de memoria se necesitan $\log_2 2^{16} = 16$ bits. Por tanto, el número de bits es 16.
b) Si las direcciones de memoria son de 16 bits, el computador es de 16 bits y los registros tendrán también 16 bits.
c) En cada posición de memoria se almacena un byte, porque el computador direcciona la memoria por bytes.
d) Si el computador fuera de 32 bits se podrían direccionar 2^{32} bytes = $2^{32} / 2^{20} = 2^{12}$ MB.
e) Como el computador tiene 100 registros se necesitan 7 bits para especificar el número de un registro. Con 6 bits solo se podrían identificar $2^6 = 64$ registros, y no sería suficiente.

Ejercicio 2. Indique el valor decimal correspondiente al siguiente número representado en formato IEEE754 **0xC1860000**

Solución:

0xC1860000 = 1100 0001 1000 0110 0000….. 00000

- El bit de signo es 1, por tanto el número es negativo
- El exponente almacenado es $10000011_{(2} = (128+3)_{(10} = 131$. El exponente real es 131 – 127 (sesgo) = 4.
- La mantisa almacenada es 00001100000. Si se añade el bit implícito, la mantisa real es 1,000011000.0000.

El numero es -1,000011 x 2^4 = -10000,11 = $-16,75_{(10}$

Ejercicio 3. Considere que el registro $a0 contiene la dirección de comienzo de una cadena de caracteres y $a1 contiene la dirección de memoria de una zona con espacio para almacenar la cadena de caracteres anterior. Escriba un fragmento de programa que copie el contenido de la primera cadena en la segunda.

Solución:

Una posible solución es:

```
bucle:    lbu  $t0, ($a0)
          beqz $t0, fin
          sb   $t0, ($a1)
          addi $a0, $a0, 1
          addi $a1, $a1, 1
          b bucle
fin:
```

Tenga en cuenta que una cadena de caracteres es un vector de caracteres (cada carácter ocupa un byte) que finaliza con el código ASCII 0 (00000000). Se utiliza la instrucción `lbu`, para leer los bytes de memoria sin signo. Los caracteres codificados en código ASCII no tienen signo.

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
2 de octubre de 2019. Grupo 82. Mini-examen 1 A

Ejercicio 1. Se quiere diseñar un procesador con 32 registros y un ancho de palabra que permita direccionar como mucho 1 MB de memoria. La memoria se direcciona a nivel de byte y los números se representan en complemento a dos. Conteste de forma breve a las siguientes preguntas.

a) ¿Qué es el contador de programa e indique de forma razonada cuántos bits tiene?
b) Indique el ancho de palabra de este computador.
c) ¿Cuántos bits se utilizan para identificar los registros de este computador?
d) Represente el número 18 en complemento a dos en este computador.

Solución:

a) Este computador puede direccionar como mucho 1MB = 2^20 bytes. Por tanto, las direcciones de memoria ocupan 20 bits y el computador por tanto es de 20 bits. El registro contador de programa deberá tener capacidad para almacenar 20 bits.
b) Este computador tiene un ancho de palabra de 20 bits, puesto que las direcciones de memoria ocupan 20 bits.
c) Como el computador tiene 32 registros se necesitan $\log_2 32 = 5$ bits.
d) Como el número es positivo, en complemento a 2 el número 18 se representa directamente en binario utilizando 20 bits: 00000000000000010010

Ejercicio 2. Indique el valor decimal correspondiente al siguiente número representado en formato IEEE754. **0x41E40000**

Solución:

0x41E40000 = 0 10000011 11001000000000

- El bit de signo es 0, por tanto, el número es positivo.
- El exponente almacenado (8 siguientes bits) es 10000011 = (128+3) = 131. El exponente real es 131 – 127 (sesgo) = 4
- La mantisa almacenada es 11001…0000. La mantisa real, añadiendo el bit implícito es 1,11001.

El valor del número es $1{,}11001 \times 2^4 = 11100{,}1_{(2} = 26{,}5$

Ejercicio 3. Dada el siguiente código en un lenguaje de alto nivel:

```
int vec[100];   # variales globales
int b = 0;

main ()
{
      int i = 0;
      for (i = 0; i < 100; i++)
            a = a + vec[i];
}
```

Se pide: escriba su equivalente en lenguaje ensamblador MIPS32.

Solución:

Una posible solución es:

```
.data:    align 2
          vec: .space    400  # vector de enteros, 400 bytes
          b:   .word      0

.text:

          li   $t0, 0         # índice i
          li   $t1, 0         # desplazamiento para el vector
          li   $t2, 100
          li   $v0, 0         # para la variable  a
   bucle: bge  $t0, $t2, fin
          lw   $t3, vec($t1)
          add  $v0, $v0, $t3
          addi $t0, $t0, 1    # se incrementa i
          addi $t1, $t1, 4    # el desplazamiento de 4 en 4 (int)
          b    bucle
     fin:
```

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
2 de octubre de 2019. Grupo 82. Mini-examen 1 B

Ejercicio 1. Se quiere diseñar un computador de 64 registros capaz de direccionar como mucho 64 KB de memoria principal, teniendo en cuenta que la memoria se direcciona por bytes. Conteste correctamente y de forma breve a las siguientes preguntas:

a) ¿Cuál es el tamaño de la palabra en este computador?
b) ¿Cuántas palabras se necesita en este computador para almacenar un número en coma flotante de doble precisión en formato IEEE 754?
c) ¿Qué es el registro de instrucción y para qué se utiliza? ¿Cuál es su tamaño?
d) ¿Cuántos bits se utilizan para identificar los registros de este computador?

Solución:

a) Como este computador puede direccionar como mucho 64 KB = 2^{16} bytes, el número de bits empleado para almacenar las direcciones de memoria es de 16 bits y, por tanto, el ancho de palabra del computador es de 16 bits.
b) Un número de coma flotante de doble precisión se representa utilizando 64 bits. En este computador se necesitarán 64 / 16 = 4 palabras.
c) El registro de instrucción es el registro donde se almacena la instrucción que está ejecutando en cada momento la unidad de control. Su tamaño coincide con el resto de los registros del computador, que coincide con el ancho de palabra, en este caso 16 bits.
d) Se necesitan $\log_2 64 = 6$ bits.

Ejercicio 2. Represente en el estándar IEEE 754 de simple precisión el valor decimal -20,5. Exprese el resultado final obligatoriamente en hexadecimal.

Solución:

$-20{,}5_{(10} = -10100{,}1_{(2} = -1{,}0100 \times 2^4$ (de forma normalizada). El bit de singo es 1 (el número es negativo). El valor del exponente es 4 + 127 = 131 = 10000011. La mantisa que se almacena (no se almacena el bit implícito) es 010000…..0000.

El número es 1100 0001 1010 0….00000 = 0xC1A00000

Ejercicio 3. Considere el siguiente fragmento de programa:

```
.data:
    Cadena: .asciiz “Este es un programa en ensamblador”
```

Escriba un fragmento de programa que cambie todas las ocurrencias del carácter ‘a’ por el carácter ‘x’. Imprima a continuación la cadena resultante.

Solución:

```
            li   $t0, ‘a’
            li   $t1, ‘x’
            la   $t2, cadena    # t2 almacena la dirección
 bucle:     lbu  $t3, ($t2)     # se lee un elemento de la cadena
            beqz $t3, fin       # si código ASCCII 9 fin
            bneq $t3, $t0, noigual
            sb   $t1, ($t2)
noigual:    addi $t2, $t2, 1
            b    bucle:
    fin:
```

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
25 de octubre de 2019 Grupo 81 Modelo A Ex 2

Ejercicio 1. Considere un computador de 16 bits que direcciona la memoria por bytes, que dispone de 32 registros y un juego de 127 instrucciones máquina. Dada la siguiente instrucción máquina ADD R1 n(R2) que suma el contenido del registro R1 con el dato almacenado en la dirección de memoria especificada por n(R2), siendo n un número entero, y deja el resultado de nuevo en R1. Se pide:

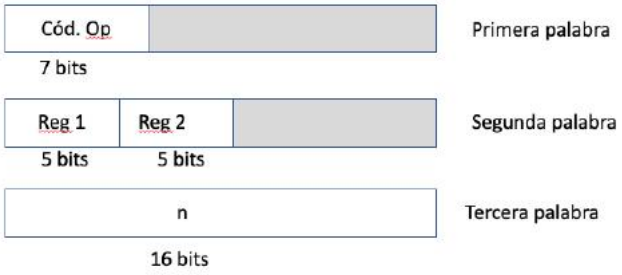
a) Indique los modos de direccionamiento que aparecen en esta instrucción:
b) Indique un formato válido para la instrucción anterior.

Solución:

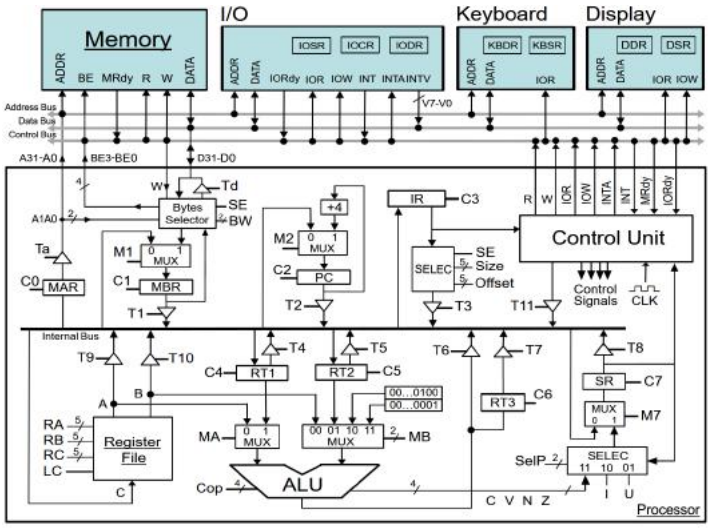
a) El campo R1 utiliza modo de direccionamiento de registro. Los campos n(R2) utiliza modo de direccionamiento relativo a registro.
b) El número de bits necesarios para codificar cada campo es:

- Código de operación: 7 bits. Hay 127 instrucciones.
- Para los campos R1 y R2 se necesitan 5 bits. Hay 32 registros en este computador.
- El número n necesita 16 bits en un computador de 16 bits.

Por tanto, se necesitan 3 palabras con el siguiente formato:



Ejercicio 2. Dado el procesador:



Indique las operaciones elementales que se ejecutan en los siguientes ciclos cuando se activan las siguientes señales de control:

Ciclo1: RA=00001, MB = 01, T6, C1
Cop = código de OR, Sep=11, M7, C7, LC

Ciclo 2: M1, C1, Ta, R, BW=11

Solución:

Ciclo 1: MBR ← R1 or RT2
Ciclo 2: MBR ← MP

Ejercicio 3. Dado las siguientes funciones en lenguaje C:

```
int F1 (int a, int b, int v[], int N, int X)
{
       int ret[10] ;
       int k;

       ret[2]= v[1] + a + X;
       k = F2(ret[2], X);

       return (k);
}
```

```
int F2 (int X, int Y )
{
       retrun (x+Y+2);

}
```

Codifique en ensamblador (tal y como está escrita) el código correspondiente a la función F1 (solo la función F1).

Solución: Una posible solución es:

```
F1:  addu $sp, $sp, -44  # espacio para ra y el vector ret
     sw   $ra, 40($sp)

     lw   $t0, 4($a2)    # v[1]

     add  $t0, $t0, $a0  # v[1] + a
     lw   $t1, 44($sp)   # x
     add  $t0, $t0, $t1  # v[1] + a + X
     sw   $t0, 8($sp)    # se almacena en ret[2]
     move $a0, $t0
     move $a1, $t1
     jal  F2             # el resultado se devuelve en $v0
     lw   $ra, 40($sp)
     addu $sp, $sp, 44
     j4   $ra
```

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
25 de octubre de 2019 **Grupo 81 Modelo B Ex 2**

Ejercicio 1. Dada la siguiente función, escriba su equivalente en ensamblador:

```
int F(int A[], int B[], int C[], int N, int X, int Y){   // A, B y C vectores de
                                                          // N componentes
      int V[10];

      V[2] = A[1] + B[2]+ C[3] + X + Y;
      return V[2];
}
```

Solución: Una posible solución es:

```
      addu $sp, $sp, -40  # se crea el vector local

      lw   $t0, 4($a0)    # A[1]
      lw   $t1, 8($a1)    # B[2]
      lw   $t1, 12($a2)   # C[3]

      add  $v0, $t0, $t1
      add  $v0, $v0, $t2  # A[1]+ B[2]+ C[3]

      lw   $t0, 40($sp)       #x
      add  $v0, $v0, $t0
      lw   $t0, 44($sp)       # y
      add  $v0, $v0, $t0
      sw   $v0, 8($sp)    # se almacena en v[2]

      addu $sp, $sp, 40
      jr   $ra
```

Ejercicio 2. Considere un computador de 32 bits que direcciona la memoria por bytes, que dispone de 64 registros y un juego de 136 instrucciones máquina. Dada la siguiente instrucción máquina CMP R1, dir, n(R2) que compara el contenido de la posición de memoria dir con el contenido de la posición especificado por n(R2). La instrucción almacena en R1 un 0 si son distintos y 1 si son iguales. Se pide:

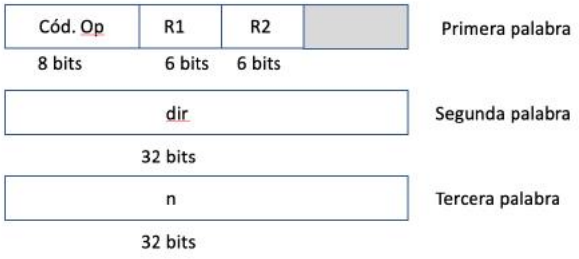
a) Indique los modos de direccionamiento que aparecen en esta instrucción:
b) Indique un formato válido para la instrucción anterior.

Solución:

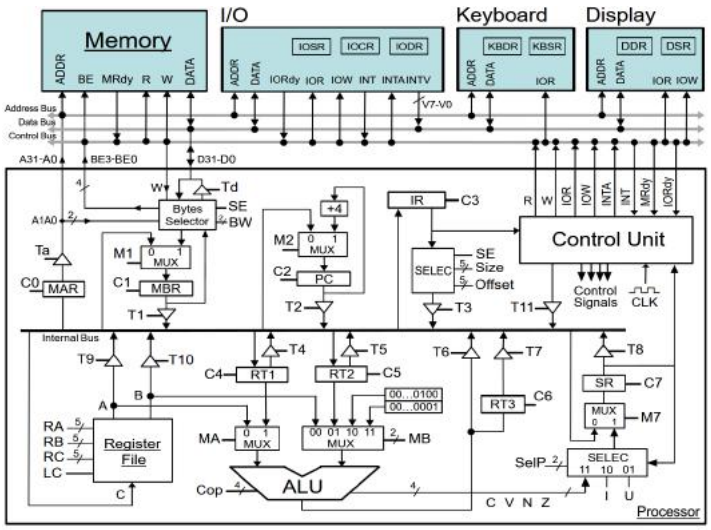
c) El campo R1 utiliza modo de direccionamiento de registro. El campo dir utiliza modo de direccionamiento directo a memoria. Los campos n(R2) utiliza modo de direccionamiento relativo a registro.
d) El número de bits necesarios para codificar cada campo es:-

- Código de operación: 8 bits. Hay 136 instrucciones.
- Para los campos R1 y R2 se necesitan 6 bits. Hay 64 registros en este computador.
- La dirección dir necesita 32 bits en un computador de 32 bits.
- El número n necesita 32 bits en un computador de 32 bits

Por tanto, se necesitan 3 palabras con el siguiente formato:



Ejercicio 3. Dado el procesador:



Indique las operaciones elementales que se ejecutan en los siguientes ciclos cuando se activan las siguientes señales de control:

Ciclo1: RB = 00010, T10, C0
Ciclo 2: MA, MB = 10, cop = OR binario, C6
Ciclo 3: T7, RC = 00100, LC

Solución:

Ciclo 1: MAR ← R2
Ciclo 2: RT3 ← RT1 or “4”
Ciclo 3: R4 ← RT3

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
22 de octubre de 2019 Grupo 82 Modelo A Ex 2

Ejercicio 1. Considere un computador de 32 bits que direcciona la memoria por bytes, que dispone de 32 registros y un juego de 136 instrucciones máquina. Dada la siguiente instrucción máquina ADD n(R1) R2 R3 que suma el contenido del registro R1 con el de R2 y deja el resultado en la dirección de memoria especificada por n(R1), siendo n un número entero. Se pide:

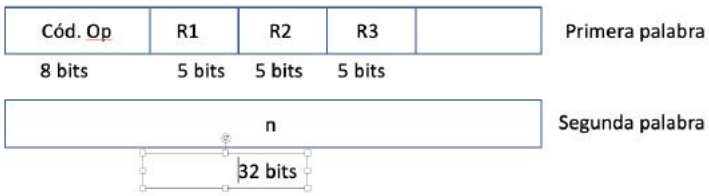
a) Indique los modos de direccionamiento que aparecen en esta instrucción:
b) Indique un formato válido para la instrucción anterior.

Solución:

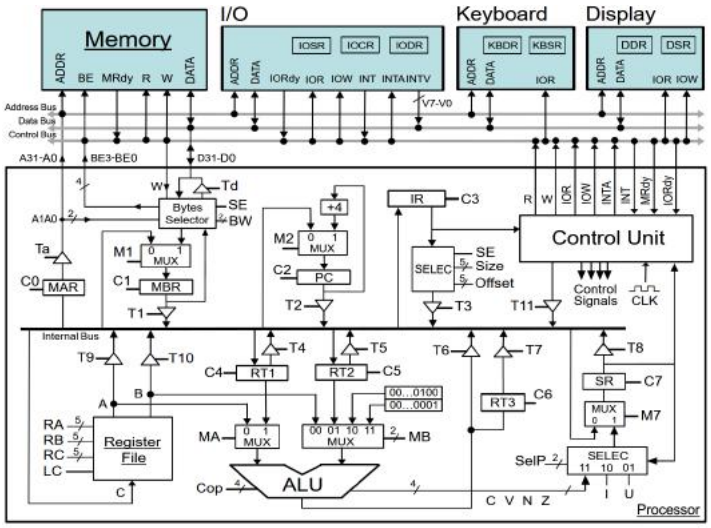
a) Los campos R2 y R3 utilizan modo de direccionamiento de registro. El campo n(R1) utiliza modo de direccionamiento relativo a registro.
b) El número de bits necesarios para codificar cada campo es:

- Código de operación: 8 bits. Hay 136 instrucciones.
- Para los campos R1, R2 y R3 se necesitan 5 bits para cada uno. Hay 32 registros en este computador.
- El número n necesita 32 bits en un computador de 32 bits.

Por tanto, se necesitan 2 palabras con el siguiente formato:



Ejercicio 2. Dado el procesador:



Indique las operaciones elementales que se ejecutan en los siguientes ciclos cuando se activan las siguientes señales de control:

Ciclo1: RA=00001, RC=00010, LC, MB = 01, T6,
Cop = código de SUMA, Sep=11, M7, C7, LC
Ciclo 2: M1, C1, Ta, R, BW=11

Solución:

Ciclo 1: R2 ← R1 + RT2
Ciclo 2: MBR ← MP[MAR]

Ejercicio 3. Dado el siguiente fragmento de programa:

```
.data
   A:   .word 1, 2, 3, 4
   B:   .float 1.2, 0.6, 3.4, 5
   C:   .word 1, 0, 0, 0
   N:   4
   X:   8
   Y:   7
```

Escriba un fragmento de programa en ensamblador que permite invocar a una función F de 6 argumentos: int F (int M1[], float M2[], int M3[], int F, int C, int H) de la siguiente manera: F(A, B, C, N, X, Y*2); a continuación se debe imprimir el resultado que devuelve la función.

Solución. Una posible solución es:

```
la   $a0, A
la   $a1, B
la   $a2, C
lw   $a3, N
lw   $t2, X

lw   $t2, Y
muli $t2, $t2, 2
addu $sp, $sp, -4
sw   $t2, ($sp)     # primero Y*2 en la pila

lw   $t2, X
addu $sp, $sp, -4
sw   $t2, ($sp)     # X en la pila

jal  F

addu $sp, $sp, 8    # restaura la pila
move $a0, $v0
li   $v0, 1
syscall
```

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
22 de octubre de 2019 **Grupo 82 Modelo B ex 2**

Ejercicio 1. Dada la siguiente función, escriba su equivalente en ensamblador:

```
int F(int A[], int B[], int C[], int N, int X, int Y){   // A, B y C vectores de
                                                          // N componentes
      int r;

      r = A[0] + B[1] + C[1] + X + Y;
      return r;
}
```

Solución. Una posible solución es:

```
        lw   $v0, ($a0)     # A[0]
        lw   $t1, 4($a1)    # A[0]
        lw   $t2, 4($a2)    # C[1]
        add  $v0, $t1, $v0
        add  $v0, $v0, $t2  # A[0]+ B[1]+ C[1]

        lw   $t1, ($sp)     # x
        lw   $t2, 4($sp)    # y
        add  $v0, $v0, $t1
        add  $v0, $v0, $t2

        jr   $ra
```

Ejercicio 2. Considere un computador de 32 bits que direcciona la memoria por bytes, que dispone de 64 registros y un juego de 98 instrucciones máquina. Dada la siguiente instrucción máquina CMP R1, dir, n(R2) que compara el contenido de la posición de memoria dir con el contenido de la posición especificado por n(R2). La instrucción almacena en R1 un 0 si son distintos y 1 si son iguales. Se pide:

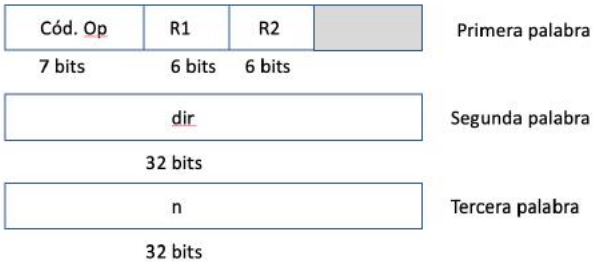
a) Indique los modos de direccionamiento que aparecen en esta instrucción:
b) Indique un formato válido para la instrucción anterior.

Solución:

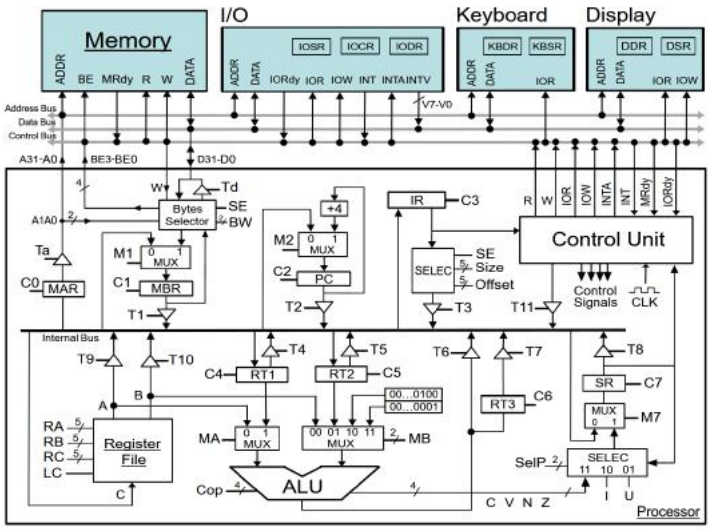
c) Los campos R1 y R2 utilizan modo de direccionamiento de registro. El campo n(R2) utiliza modo de direccionamiento relativo a registro. El campo dir utiliza modo de direccionamiento directo a memoria.
d) El número de bits necesarios para codificar cada campo es:

- Código de operación: 7 bits. Hay 98 instrucciones.
- Para los campos R1 y R2 se necesitan 6 bits para cada uno. Hay 64 registros en este computador.
- El número n necesita 32 bits en un computador de 32 bits.
- El campo dir necesita 32 bits en un computador de 32 bits.

Por tanto, se necesitan 3 palabras con el siguiente formato:



Ejercicio 3. Dado el procesador:



Indique las operaciones elementales que se ejecutan en los siguientes ciclos cuando se activan las siguientes señales de control:

Ciclo1: RB = 00010, T10, C4
Ciclo 2: MA, MB = 10, cop = OR binario, C6
Ciclo 3: T7, RC = 00100, LC

Solución:

Ciclo 1: RT1 ← R2
Ciclo 2: RT3 ← RT1 or “4”
Ciclo 3: R4 ← RT3

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
15 de noviembre de 2019. Grupo 81. Examen 3 A

Ejercicio 1. Escriba el código de la siguiente función:

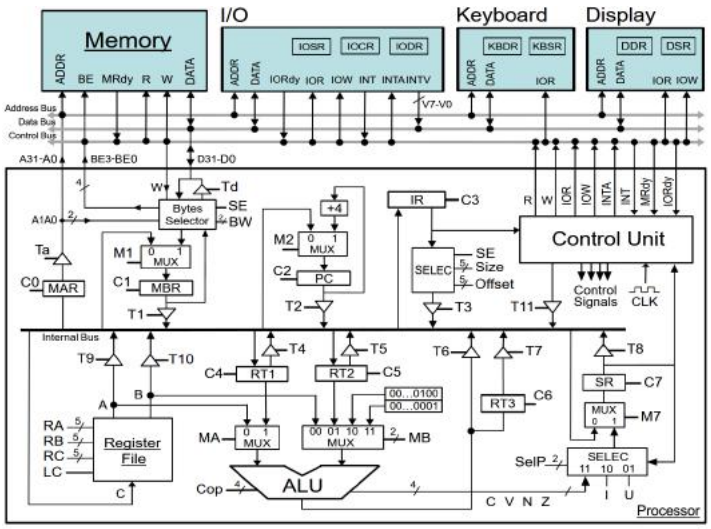
```
int EsIgual(int Mat [][], int M, int N, int i, int j, int X)
```

El primer argumento representa una matriz cuadrada de dimensiones M x N (dada por el segundo argumento y tercer argumento). La función devuelve 1 si el valor de X es igual al valor almacenado en la posición [i][j] de la matriz. Devuelve 0 en caso contario. Devuelve -1 si i o j están fuera de rango.

Solución:

```
            lw   $t0, ($sp)      # j
            lw   $t1, 4($sp)     # x

            blt  $a3, $0, error
            bge  $a3, $a1, error
            blt  $t0, $0, error
            bge  $t0, $a2, error

            #acceso a Mat[i][j] ->  a0+(ix4xN) + jx4
            muli $t5, $a2, 4     # 4 x N
            mul  $t5, $t5, $a3   # 4* N * i
            mul  $t6, $t0, 4     # j x 4
            add  $t5, $t5, $t6
            add  $t5, $a0, $t6   # dirección del elemento (i,j)

            lw   $t5, ($t5)
            beq  $t5, $t1, iguales
            li   $v0, 0
            jr   $ra        # no son iguales

iguales:    li   $v0, 1
            jr   $ra

error:      li   $v0, -1
            jr   $ra
```

Ejercicio2. Dada la estructura del computador visto en clase. Indique operaciones elementales y señales de control necesarios para ejecutar la hipotética instrucción `ADDSP R1 R2` Esta instrucción suma el contenido de los registros R1 y R2 y almacena el resultado en la cima de la pila (haciendo hueco en la misma). Incluya el código de *fetch* y asuma que la memoria necesita 1 ciclo para las operaciones de lectura y escritura y un ciclo para la decodificación. El registro puntero de pila es el registro 29 del banco de registros.



Solución:

Ciclo	Operación Elemental	Señales de control
C0	MAR ← PC	T2, C0
C1	MBR ← MP PC ← PC + 4	TD, BW=11, R, M1, C1, M2,C2
C2	IR ← MBR	T1, C3
C3	DECODIFICACIÓN	
C4	MBR ← R1 + R2	RA=00001, RB=00010, T6, C1, COP=SUMA, SELP=11, M7, C7
C5	SP ← SP -4 MAR ← SP -4	RA=29, RC=29, MB=10, COP=RESTA, T6, LC, C0
C6	MP[MAR] ← MBR	W, BW=11, TD, TA

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
15 de noviembre de 2019. Grupo 81. Examen 3 B

Ejercicio 1. Escriba el código de la siguiente función:

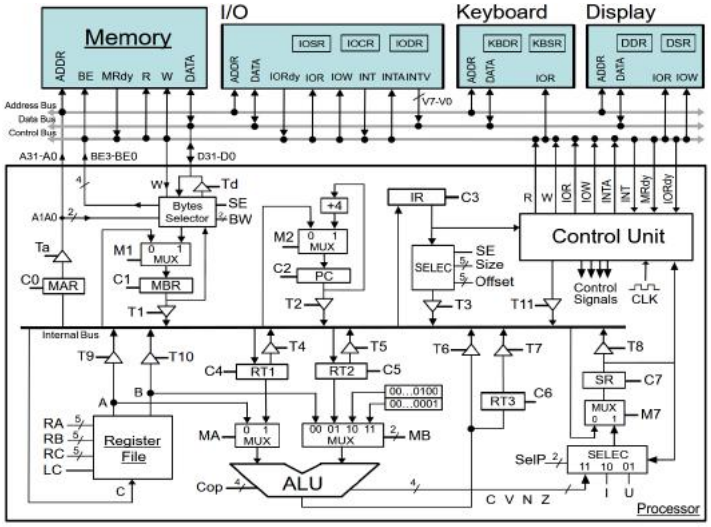
```
int intercambia(char A[][], int N, int i, int j)
```

Esta función recibe una matriz A de dimensión `NxN`. Si `i` o `j` están fuera de rango, la función devuelve -1 y no realiza ninguna función. En caso contrario la función intercambia el contenido de las filas `i` y `j` y devuelve 0.

Solución:

```
Intercambia:    blt   $a2, $0, error
                bge   $a2, $a1,error

                blt   $a3, $0, error
                bge   $a3, $a1,error


                # calcula dirección fila i, cada elemento ocupa un byte
                muli  $t0, $a2, $a1
                add   $t0, $a0, $t0   # inicio de fila i

                # calcula dirección fila j, cada elemento ocupa un byte
                muli  $t1, $a1, $a3
                add   $t1, $a0, $t1   # inicio de fila i

                li    $t3, 0     # índice del bucle

    bucle:      bge   $t3, $a1, fin
                lb    $t5, ($t0)
                lb    $t6, ($t1)
                sb    $t5, ($t1)
                sb    $t6, ($t0)

                addi  $t3, $t3, 1
                addi  $t0, $t0, 1
                addi  $t1, $t1, 1

                b     bucle
    fin:        li    $v0, 0
                jr    $ra

    error:      li    $v0, -1
                jr    $ra
```

Ejercicio2. Considere la estructura del computador visto en clase y la hipotética instrucción, que ocupa dos palabras, `POP dirección.` Esta instrucción extrae de la pila el dato almacenado en la cima de la pila y lo almacena en la posición de memoria dada por `dirección`. (Registro puntero de pila = R29). Indique los ciclos y operaciones elementales necesarios para ejecutar la instrucción anterior. Incluya el ciclo de *fetch* y asuma que la memoria necesita 1 ciclo para las operaciones de lectura y escritura y un ciclo para la decodificación.



Solución:

Ciclo	Operación Elemental	Señales de control
C0	MAR ← PC	T2, C0
C1	MBR ← MP PC ← PC + 4	TD, BW=11, R, M1, C1, M2,C2
C2	IR ← MBR	T1, C3
C3	DECODIFICACIÓN	
C4	MAR ← SP	RA=29, T0, C0
C5	MBR ← MP SP ← SP + 4	R, Td, M1, C1, BW=11 RA=RB=29, MB=10, COP=SUMA, LC
C6	RT1 ← MBR	T1, C4
C7	MAR ← PC	C0, T2
C8	MBR ← MP PC ← PC + 4	R, Td, M1, C1, BW=11 M2, C2
C9	MAR ← MBR	T1, C0
C10	MBR ← RT1	T4, C1
C11	MP ← MBR	W, BW=11, Td, Ta

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
13 de noviembre de 2019. Grupo 82. Examen 3 A

Ejercicio 1. Escriba el código de la siguiente función:

```
int sumar(int Mat [][], int N, int i, int j, int k, int l)
```

El primer argumento representa una matriz cuadrada de dimensión N (dada por el segundo argumento). La función suma el elemento Mat[i][j] con el elemento Mat[k][l] y devuelve su resultado.

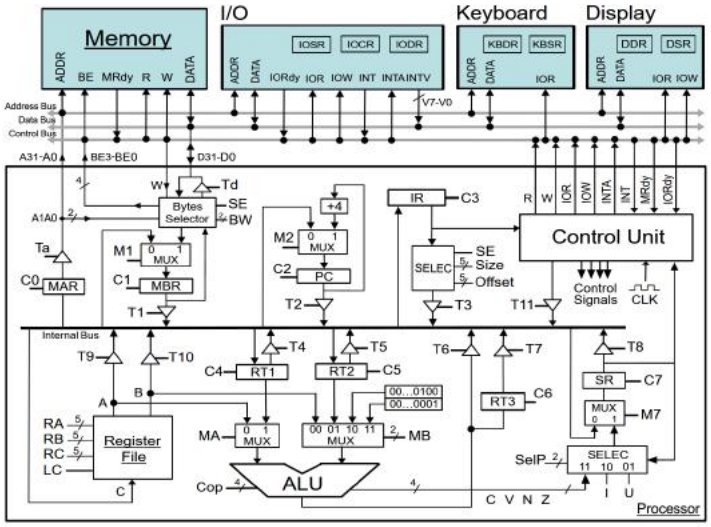
Solución:

```
Sumar:          lw   $t0, ($sp)      #k
                lw   $t1, 4($sp)     #l

                # acceso a mat[i][j]
                muli $t3, $a1, 4     # 4 x N
                mul  $t4, $t3, $a2   # 4 x N x i
                muli $t5, $a3, 4     # 4 X j
                add  $t4, $t4, $t5   # 4 x N x i + 4 x j
                add  $t4, $t4, $a0   # dirección de (i,j)
                lw   $v0, ($t4)      # elemento (i,j)

                # acceso a mat[k][l]
                muli $t3, $a1, 4     # 4 x N
                mul  $t4, $t3, $t0   # 4 x N x k
                muli $t5, $t1, 4     # 4 X l
                add  $t4, $t4, $t5   # 4 x N x k + 4 x l
                add  $t4, $t4, $a0   # dirección de (k,l)
                lw   $v1, ($t4)      # elemento (k,l)

                add  $v0, $v0, $v1
                jr   $ra
```

Ejercicio2. Dada la estructura del computador visto en clase. Se pide:

a) Indique operaciones elementales y señales de control necesarios para ejecutar la instrucción del MIPS: `lw $t1,n($t2).` Incluya el código de *fetch* y asuma que la memoria necesita 1 ciclo para las operaciones de lectura y escritura y un ciclo para la decodificación.

b) ¿Se podría reducir el número de ciclos con algún cambio en la estructura del computador? Indique cuál.



Solución:

Ciclo	Operación Elemental	Señales de control
C0	MAR ← PC	T2, C0
C1	MBR ← MP PC ← PC + 4	TD, BW=11, R, M1, C1, M2,C2
C2	IR ← MBR	T1, C3
C3	DECODIFICACIÓN	
C4	RT1 ← IR(n)	SE=1, SIZE=16, OFFSET=0, T3, C4
C5	MAR ← RT1 + $t2	RB = $t2, MA, T6, C0
C6	MBR ← MP	R, BW=11, Td, M1, C1
C7	$t1 ← MBR	T1, RC=$t1, LC

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
12 de noviembre de 2019. Grupo 82. Examen 3 B

Ejercicio 1. Escriba el código de la siguiente función:

```
int trasponer(char A[][], char B[][], int N, int M)
```

La matriz A es de dimensión NxM y la matriz B es de dimensión MxN. La función almacena en la matriz B la traspuesta de la matriz A.

Solución:

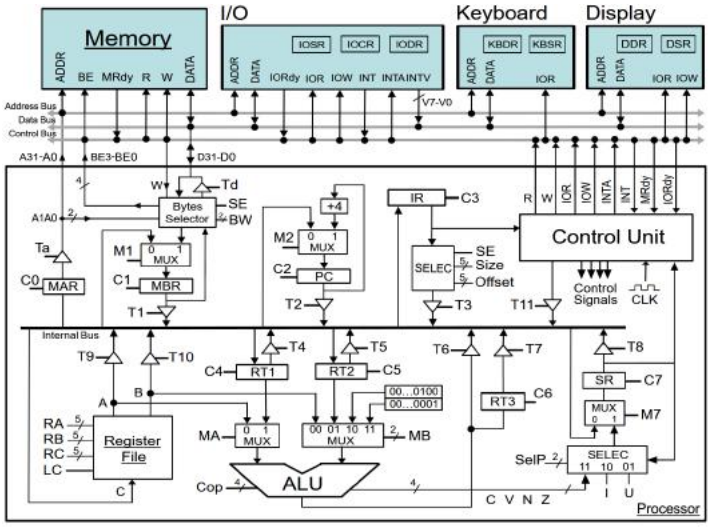
Un pseudocódigo para esta función es el siguiente:

```
for (i=0; i <N; i++)
     for (j=0; j<M; j++)
          B[j][i] = A[i][j];
```

```
trasponer:      li   $t0, 0     # i
                li   $t1,  0    # j

     bucle1:    bge  $t0, $a2, fin1
                li   $t1, 0     # j=0

     bucle2:    bge  $t1, $a3, fin2
                # acceso a A[i][j]
                muli $t3, $t0, 4     # i x 4
                mul  $t3, $t3, $a2   # i x 4 x N
                muli $t4, $t1, 4     # j x 4
                add  $t3, $t3, $t4
                add  $t5, $a0, $t3
                lb   $t6, ($t5)      # elemento (i,j)

                # acceso a B[j][i]
                muli $t3, $t1, 4     # j x 4
                mul  $t3, $t3, $a3   # j x 4 x M
                muli $t4, $t0, 4     # i x 4
                add  $t3, $t3, $t4
                add  $t5, $a1, $t3
                sb   $t6, ($t5)

                addi $t1, $t1, 1
                b bucle2
     fin2:      addi $t0, $t0, 1
                b    bucle1
     fin1:      jr   $ra
```

Ejercicio2. Considere la estructura del computador visto en clase y la hipotética instrucción, que ocupa dos palabras, `SAWP dirección.` Esta instrucción intercambia el contenido de la posición de memoria almacenado en `dirección` con el contenido almacenado en la cima de la pila (Registro puntero de pila = R29). Se pide:

a) Indique los ciclos y operaciones elementales necesarios para ejecutar la instrucción anterior. Incluya el ciclo de *fetch* y asuma que la memoria necesita 1 ciclo para las operaciones de lectura y escritura y un ciclo para la decodificación.



Solución:

Ciclo	Operación Elemental	Señales de control
C0	MAR ← PC	T2, C0
C1	MBR ← MP PC ← PC + 4	TD, BW=11, R, M1, C1, M2,C2
C2	IR ← MBR	T1, C3
C3	DECODIFICACIÓN	
C4	MAR ← SP	RA=29, T9, C0
C5	MBR ← MP	R, BW=11, TD, M1, C1
C6	RT1 ← MBR (dato de la cima de la pila)	C4, T1
C7	MAR ← PC	T2, C0
C8	MBR ← MP PC ← PC + 4	R, BW=11, TD, M1, C1 M2, C2
C9	MAR ← MBR	T1, C0
C10	MBR ← MP	R, BW=11, TD, M1, C1
C11	RT2←MBR (dato almacenado en dirección)	T1, C5
C12	MBR←RT1	T4, C1
C13	MBR ← MP	R, BW011, TD, M1, C1
C14	MBR←RT2	T5, C1
C15	MAR ← SP	RA=29, T9, C0
C16	MP ← MBR	W, BW=11, TA, TD

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
13 de diciembre de 2019. Grupo 81. Mini-examen 4 A

Ejercicio 1 (1 punto). ¿Qué es el contador de programa e indique cómo se relaciona con el ancho de palabra de un computador?

Solución:

El contador de programa es el registro del procesador que almacena la dirección de la siguiente instrucción a ejecutar. Su tamaño coincide con el ancho de palabra de un computador.

Ejercicio 2 (1 punto). Considere un disco duro de 7200 revoluciones por minuto, con un tiempo medio de búsqueda de 2 ms y 1000 sectores por pista. Calcule el tiempo de acceso a un sector.

Solución:

Tacceso= Tbusqueda + Trotacion + Ttransf = 2 +1/2 (60000/7200) + (60000/7200)*(1/1000) = 2 + 4,17 + 0,0083 = 6,18 ms

Ejercicio 3 (4 puntos). Considere un computador de 32 bits con una memoria caché (unificada para datos e instrucciones) de 64 KB, asociativa por conjuntos de 4 vías y líneas de 64 bytes. Considere el siguiente fragmento de programa:

```
.data:
       v1: .space 640
.text:
       li   $t0, 0
       li   $t1, 0
       li   $v0, 0
       li   $t2, 640
bucle: bge  $t0, $t2, fin
       lb   $t3,  v1($t1)
       add  $v0, $v0, $t3
       addi $t0, $t0, 1
       addi $t1, $t1, 1
       b    bucle
fin:   move $v0, $t1
```

Se pide:

a) Indique el número de líneas y conjuntos de la caché.
b) Indique la tasa de fallos que se obtiene en la ejecución del fragmento de código anterior asumiendo que la caché está inicialmente vacía.

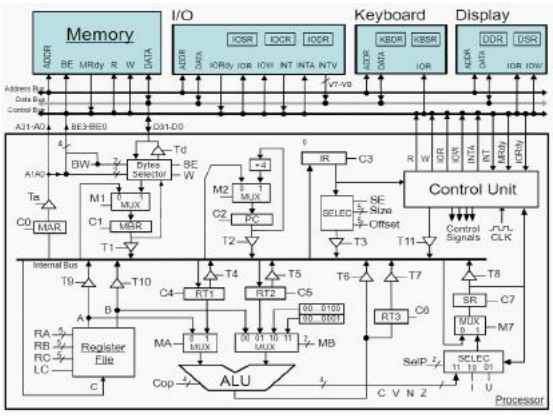
Solución:

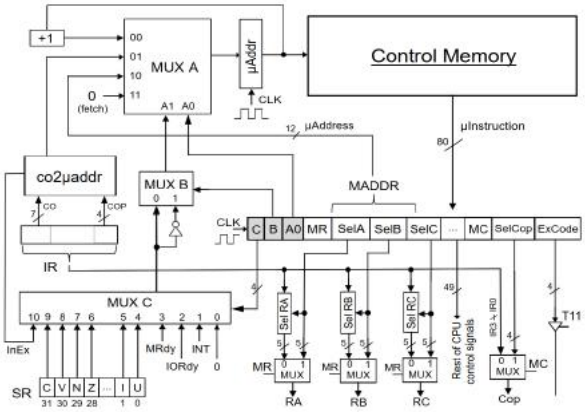
a) Número de líneas = 64 KB / 64 bytes = 2^{16} / 2^{6} = 2^{10} = 1024 líneas
Número de conjuntos = 2^{10} / 2^{2} = 2^{8} = 256 conjuntos

b) En el fragmento de programa anterior se ejecutan 4 + 6 x 640 + 2 = 3846 instrucciones. El bucle se ejecuta 6240 veces. Cada instrucción supone un acceso a memoria. Además, se produce un acceso a datos (instrucción lb). En total hay 3846 + 640 = 4486 accesos a memoria.
Se va a analizar por separado el acceso a instrucción y a datos. Cuando se accede a la primera instrucción se produce un fallo y se transfiere una línea (con instrucciones) de memoria principal a caché. En una línea caben 64 / 4 = 16 instrucciones. Por tanto, se transfiere a caché todas las instrucciones del fragmento anterior. Esto hace que en los restantes accesos a instrucciones todos los accesos sean aciertos en la caché.

En cuanto a los datos, cuando se accede al primer elemento del vector v1 se produce un fallo. El vector se trata como un vector de bytes (el registro $t1 se incrementa de 1 en 1), por tanto, se transfiere a memoria caché una línea en la que se almacenan 64 elementos del vector. Por tanto, se produce un fallo cada 64 accesos al vector. En total se producen 640/64 = 10 fallos. En total se producen (teniendo en cuenta instrucciones y datos) 11 fallos y la tasa de fallos es 11/4486.

Ejercicio 4 (4 puntos). Dado el procesador WepSIM y su unidad de control:





Especifique todas las señales de control necesarias para ejecutar las siguientes operaciones elementales:

Solución:

Reg ← RT1 (el campo Reg se encuentra en la instrucción entre los bits 25 y 21)	T4, SelC=21, LC
RT1 ← R4 (el registro 4 es implícito en la instrucción)	C4, SelA=00100, MR=1, T9
If (SR(N) == 0) salta a la microdirección que se encuentra en la etiqueta utilizada en WepSim `salto`	C= 0111, A0=0, B= 1, MADDR =salto

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
13 de diciembre de 2019. Grupo 81. Mini-examen 4 B

Ejercicio 1 (1 punto). ¿Qué es el registro de instrucción e indique cómo se relaciona con el ancho de palabra de un computador?

Solución:

El registro de instrucción es el registro del procesador que almacena la siguiente instrucción que se está ejecutando en el procesador. Su tamaño coincide con el ancho de palabra de un computador.

Ejercicio 2 (1 punto). Considere un disco duro de 5400 revoluciones por minuto, con un tiempo medio de búsqueda de 4 ms y 500 sectores por pista. Calcule el tiempo de acceso a 10 sectores consecutivos de una misma pista.

Solución:

Tacceso= Tbusqueda + Trotacion + Ttransf = 4 +1/2 (60000/5400) + 10*(60000/5400)*(1/500) = 2 + 5,55 + 0,22 = 9,77 ms. En este caso, se transfieren 10 sectores consecutivos.

Ejercicio 3 (4 puntos). Considere un computador de 32 bits con una memoria caché (unificada para datos e instrucciones) de 128 KB, asociativa por conjuntos de 8 vías y líneas de 128 bytes. Considere el siguiente fragmento de programa:

```
char v1 [1000];
char v2 [1000];
int  v3 [1000];

for (i =0; i < 1000; i++) {
     v3[i] = (int) (v1[i] + v2[i]);
```

Se pide:

a) Indique el número de líneas y conjuntos de la caché.
b) Indique la tasa de fallos a datos que se obtiene en la ejecución del fragmento de código anterior asumiendo que la caché está inicialmente vacía.

Solución:

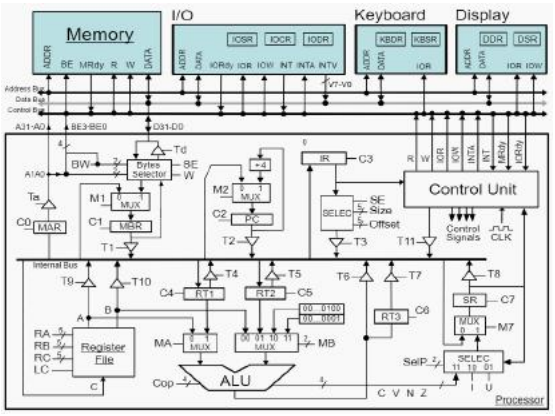
a) Número de líneas = 128 KB / 128 bytes = $2^{17} / 2^{7} = 2^{10}$ = 1024 líneas
   Número de conjuntos = $2^{10} / 2^{3} = 2^{7}$ = 128 conjuntos
b) Analizando el patrón de accesos:
   Para i = 0: se producen tres accesos a memoria:
   Read v1[0] -> fallo -> se transfiere a caché v1[0] … v1[127] # en cada línea caben 128 bytes
   Read v2[0] -> fallo -> se transfiere a caché v2[0] … v2[127] # en cada línea caben 128 bytes
   Write v3[0] -> fallo -> se transfiere a caché v3[0] … v3[31] # en cada línea caben 32 enteros

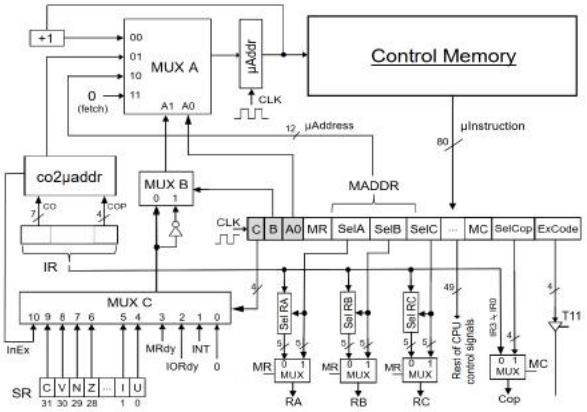
   Para i = 1 hasta i = 31 todos los accesos son aciertos.
   Para i= 32, los accesos a v1 y v2 son aciertos y el acceso a v3[32] es un fallo. Se transfiere en este caso a caché v3[32] … v3[63].

   Por tanto, cada 128 accesos se produce 1 fallo en v1, 1 fallo en v2 y 4 fallos en v3. En total 6 fallos cada 128 accesos. En total 6 * (1000/ 128) = 6 x 8 = 48 (se redondea al entero superior)

   El número de accesos a memoria de datos es 1000 x 3 = 3000 accesos. La tasa de fallos = 48/ 3000

Ejercicio 4 (4 puntos). Dado el procesador WepSIM y su unidad de control:





Especifique todas las señales de control necesarias para ejecutar las siguientes operaciones elementales:

Solución:

RT1 ← regOr1 + regOr2 (el campo RegOr1 se encuentra en la instrucción entre los bits 25 y 21 y el campo regOr2 entre los bits 10 y 6). Esta operación actualiza el registro de estado	SelA=21, SelB=6, COP=1010, T6, C4, Selp=11, M7, C7
SR ← R7 (el registro 7 es implícito en la instrucción)	SelA= 00111, MR=1, T9, C7
If (SR(V) == 1) salta a la microdirección que se encuentra en la etiqueta utilizada en WepSim `salto`	C=1000, A0=0, B=0, MADDR = salto

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
11 de diciembre de 2019. Grupo 82. Mini-examen 4 A

Ejercicio 1 (1 punto). ¿Qué es el contador de programa e indique cómo se relaciona con el ancho de palabra de un computador?

Solución:

El contador de programa es el registro del procesador que almacena la dirección de la siguiente instrucción a ejecutar. Su tamaño coincide con el ancho de palabra de un computador.

Ejercicio 2 (1 punto). Represente en el estándar IEEE 754 de simple precisión el valor decimal -16,5

Solución:

$-16{,}5_{(10} = -10000{,}1_{(2} = -1{,}0000 \times 2^4$ (de forma normalizada). El bit de singo es 1 (el número es negativo). El valor del exponente es 4 + 127 = 131 = 10000011. La mantisa que se almacena (no se almacena el bit implícito) es 000000…..0000.

El número es 1100 0001 1000 0….00000 = 0xC1800000

Ejercicio 3 (4 puntos). Considere un computador de 32 bits con una memoria caché (unificada para datos e instrucciones) de 128 KB, asociativa por conjuntos de 8 vías y líneas de 64 bytes. Considere el siguiente fragmento de programa:

```
.data:
       v1: .space 640
       v2: .space 640
.text:
       li   $t0, 0
       li   $t1, 0
       li   $t2, 640
bucle: bge  $t0, $t2, fin
       lb   $t3,  v1($t1)
       sb   $t3,  v2($t1)
       addi $t0, $t0, 1
       addi $t1, $t1, 1
       b    bucle
fin:   move $v0, $t1
```

Se pide:

a) Indique el número de líneas y conjuntos de la caché.
b) Indique la tasa de fallos que se obtiene en la ejecución del fragmento de código anterior asumiendo que la caché está inicialmente vacía.

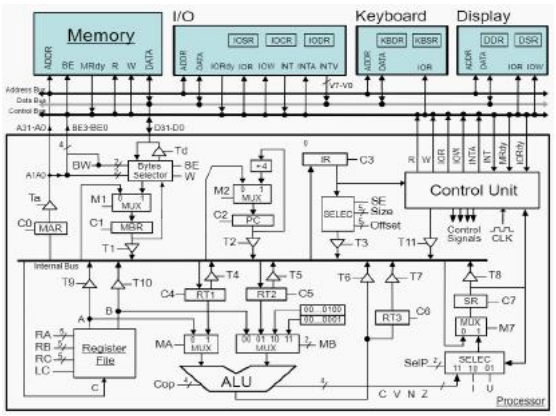
Solución:

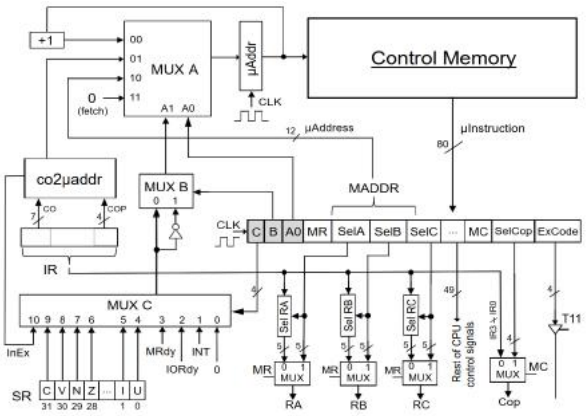
a) Número de líneas = 128 KB / 64 bytes = $2^{17} / 2^6 = 2^{11}$ = 2048 líneas
Número de conjuntos = $2^{11} / 2^3 = 2^7$ = 128 conjuntos

b) En el fragmento de programa anterior se ejecutan 3 + 6 x 640 + 2 = 3845 instrucciones. Cada instrucción supone un acceso a memoria. Además, se producen dos accesos a datos dentro del bucle (instrucción lb y sb). En total hay 3845 + 2x640 = 5125 accesos a memoria.
Se va a analizar por separado el acceso a instrucción y a datos. Cuando se accede a la primera instrucción se produce un fallo y se transfiere una línea (con instrucciones) de memoria principal a caché. En una línea caben 64 / 4 = 16 instrucciones (cada instrucción ocupa una palabra, 4 bytes). Por tanto, se transfiere a caché todas las instrucciones del fragmento anterior. Esto hace que en los restantes accesos a instrucciones todos los accesos sean aciertos en la caché.

En cuanto a los datos, cuando se accede al primer elemento del vector v1 se produce un fallo. El vector se trata como un vector de bytes (el incremento de los registros $t0 y también $t1 se hace de 1 en 1), por tanto, se transfiere a memoria caché una línea en la que se almacenan 64 elementos del vector (una línea son 64 bytes). Por tanto, se produce un fallo cada 64 accesos al vector. En total se producen 640/64 = 10 fallos por cada vector. En total se producen (teniendo en cuenta instrucciones y datos) 21 fallos y la tasa de fallos es 21/5125.

Ejercicio 4 (4 puntos). Dado el procesador WepSIM y su unidad de control:





Especifique las operaciones elementales y señales de control necesarias para ejecutar la instrucción máquina `ADD R1, R2, R3, R4`. Si el contenido del registro R1 es negativo, almacena en R2 la suma de los registros R3 y R4. En caso contario almacena en R2 un 0. Esta instrucción no modifica el registro de estado. Los campos R1, R2, R3 y R4 están consecutivos y R1 comienza en el bit (más significativo) 23. Especifique a partir de la decodificación (sin el fetch).

Solución:

Ciclo	Operación Elemental	Señales de control
C0	RT1 ← SR (se guarda temporalmente SR)	T8, C4
C1	R1 or R1	Para saber si el contenido de R1 es negativo hacemos una operación or en la ALU SelA=18, SelB=18, SelP=11, M7, C7, COP=OR
C2	If (SR(N) == 1) salto a C5	C=0111, A0 =0, B=0, MADDR=C5
C3	R2 ← R2 – R2	Se almacena 0 SelA = SelB = SelC = 13, T7, LC
C4	SR ← RT1 (se restaura SR)	T4, C7, A0=1, B=1, C=0 (salto a fetch)
C5	R2 ← R3 + R4	SelA=8, SelB=3, SelC=13, COP = ADD, LC, T6
C6	SR ← RT1 (se restaura SR)	T4, C7, A0=1, B=1, C=0 (salto a fetch)

UNIVERSIDAD CARLOS III DE MADRI
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES
11 de diciembre de 2019. Grupo 82. Mini-examen 4 B

Ejercicio 1 (1 punto). ¿Qué es el registro de instrucción e indique cómo se relaciona con el ancho de palabra de un computador?

Solución:

El registro de instrucción es el registro del procesador que almacena la siguiente instrucción que se está ejecutando en el procesador. Su tamaño coincide con el ancho de palabra de un computador.

Ejercicio 2 (1 punto). Cuál es el valor decimal del siguiente número en coma flotante de simple precisión: 0x40E00000

Solución:

0x40E0000 = 0100 0000 1110 0000 …..0000

- El bit de signo es 0, por tanto, el número es positivo.
- El exponente almacenado (8 siguientes bits) es 10000001 = (128+1) = 129. El exponente real es 129 – 127 (sesgo) = 2
- La mantisa almacenada es 110000…0000. La mantisa real, añadiendo el bit implícito es 1,1100.

El valor del número es $1{,}1100 \times 2^2 = 111_{(2} = 7$

Ejercicio 3 (4 puntos). Considere un computador de 64 bits con una memoria caché (unificada para datos e instrucciones) de 64KB, asociativa por conjuntos de 4 vías y líneas de 128 bytes. Considere el siguiente fragmento de programa:

```
.data:
       v1: .space 1024
.text:
       li   $v0, 0
       li   $t0, 0
       li   $t1, 0
       li   $t2, 128
bucle: bge  $t0, $t2, fin
       lw  $t3,  v1($t1)
       add  $v0, $v0, $t3
       addi $t0, $t0, 1
       addi $t1, $t1, 8
       b    bucle
fin:   move $v1, $v0
```

Se pide:

a) Indique el número de líneas y conjuntos de la caché.
b) Indique la tasa de fallos que se obtiene en la ejecución del fragmento de código anterior asumiendo que la caché está inicialmente vacía.

Solución:

a) Número de líneas = 64 KB / 128 bytes = $2^{16} / 2^7 = 2^9 = 512$ líneas
Número de conjuntos = $2^9 / 2^2 = 2^7 = 128$ conjuntos

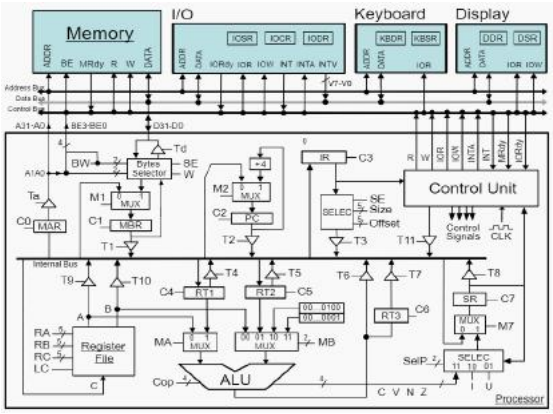
b) En el fragmento de programa anterior se ejecutan 4 + 6 x 128 + 2 = 774 instrucciones. El bucle se ejecuta 128 veces. Cada instrucción supone un acceso a memoria. Además, se produce un acceso a datos dentro del bucle (instrucción lw). En total hay 774 + 128 = 902 accesos a memoria.
Se va a analizar por separado el acceso a instrucción y a datos. Cuando se accede a la primera instrucción se produce un fallo y se transfiere una línea (con instrucciones) de memoria principal a caché. En una línea caben 128 / 4 = 32 instrucciones (cada instrucción ocupa una palabra, 4 bytes). Por tanto, se transfiere a caché todas las instrucciones del fragmento anterior. Esto hace que en los restantes accesos a instrucciones todos los accesos sean aciertos en la caché.

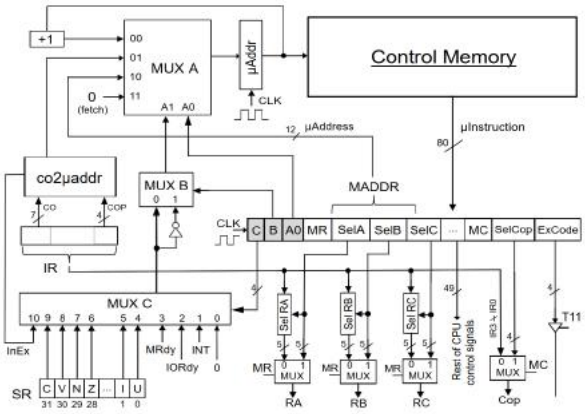
Al vector se accede utilizando la instrucción lw, que transfiere una palabra de 4 bytes. Por otra parte, el registro $t1 se incrementa de 8 en 8. Considerando que v1 se trata como un vector de enteros, el patrón de accesos a este vector es v1[0], v1[2], v1[4].

Cuando se accede a v1[0] se produce un fallo y se transfiere una línea a caché. En una línea de 128 bytes se almacenan 128/4 = 32 enteros. Para el primer caso se transfiere v1[0]..v1[32]. De todos estos datos, solo se accede posteriormente a los elementos con índice par. Es decir, se accede a solo 16 elementos de los 32 que se han traído desde memoria principal a caché. Por tanto, cada 16 iteraciones, hay un fallo. Como el bucle se ejecuta 128 veces, el número de fallos que se produce es 128/16=8.

El número total de fallos (instrucciones más datos) = 1 + 8 = 9. La tasa de fallos es 9/902

Ejercicio 4 (4 puntos). Dado el procesador WepSIM y su unidad de control:





Especifique las operaciones elementales y señales de control necesarias para ejecutar la instrucción máquina `SLT R1, R2, R3`. Si el contenido del registro R2 es menor que R3 almacena en R1 un 1, en caso contrario un 0. Esta instrucción no modifica el registro de estado. Los campos R1, R2 y R3 están consecutivos y R1 comienza en el bit (más significativo) 25. Especifique las operaciones y señales a partir de la decodificación (sin el fetch).

Solución:

Ciclo	Operación Elemental	Señales de control
C0	RT1 ← SR (se guarda SR temporalmente)	T8, C4
C1	R2-R3	Para comprobar si R2 <R3 se hace la resta y se modifica el registro de estado. R2 será menor si el bit N (negativo) se ha activado SelA=16, SelB=11, COP=RESTA, SelP=11, M7, C7
C2	If (SR(N)==1) salto a C5	A0=0, B=0, C=0111, MADDR = C5
C3	R1 ← R1 – R1 (se almacena 0)	SelA=SelB=SelC= 21, COP =RESTA, T6, LC
C4	SR ← RT1 (se restaura SR)	T4, C7, A0=1, B=1, C=0 (salto a fetch)
C5	RT2 ← R0 + 1 (se almacena 1)	SelA=0, MR, MB=10, COP=SUMA, T6, C5
C6	R1 ← RT2	SelC=21, T5, LC
C7	SR ← RT1 (se restaura SR)	T4, C7, A0=1, B=1, C=0 (salto a fetch)