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

IIC2343 – Arquitectura de Computadores (II/2014)

Solución Interrogación 2

Pregunta 1

a) Describa una convención de llamada para x86, que sea más rápida que *stdcall* al momento de leer y escribir parámetros y valores de retorno. Contrapese las posibles ventajas y desventajas **(1 pto.)**
Solución: Una posible convención podría utilizar los registros de la arquitectura x86 para almacenar los parámetros de entrada de las subrutinas. Si la cantidad de parámetros excede la cantidad de registros, se utiliza el stack para los restantes parámetros. Comparada con *stdcall*, esta convención es más rapida para leer los argumentos, pero también es más compleja, ya que no todos se ubican en el mismo lugar.

b) ¿Cuál es la función del clock en un computador? ¿Que pasaría si este es removido? **(1 pto.)**
Solución: El clock permite sincronizar todos los elementos de memoria de un computador. Sin él, no existe garantía de que los resultados almacenados en los elementos de memoria, luego de cualquier operación, sean correctos.

c) Dada la microarquitectura del computador básico, ¿es posible crear una **ISA** distinta la actual? Argumente su respuesta. **(1 pto.)**
Solución: Es posible crear nuevas ISAs usando un subconjunto de las instrucciones soportadas por la ISA actual. Además, es posible utilizar también un superconjunto, utilizando instrucciones que no tienen un *opcode* asignado, pero que tienen una combinación de señales de control.

d) ¿Por qué muchos de los dispositivos móviles, como smartphones y tablets, utilizan arquitecturas basadas en **RISC**? Argumente su respuesta. **(1 pto.)**
Solución: La arquitectura RISC, al estar basada en pocas instrucciones muy básicas, utiliza hardware simple que requiere poca energía. Esto la hace una arquitectura idea para dispositivos donde el consumo de energía es un parámetro crítico.

e) Escriba un programa en assembly del computador básico, que calcule el inverso aditivo de un número entero de 8 bits. **(1 pto.)**
Solución:

```
DATA
n         ?
n_inv

CODE
MOV A, (n)
NOT B,A
INC B
MOV (n_inv), B
```

f) ¿Es posible emular el funcionamiento del registro **BP** en el computador básico, sin modificar la arquitectura? Si su respuesta es positiva, esboce la solución. Si es negativa, explique el motivo. **(1 pto.)**
Solución: Si es posible emular el registro BP, utilizando una dirección fija de memoria. De esta manera, en vez de hacer referencia **BP**, se hará referecia a esta dirección. El único cuidado necesario es asegurarse que el valor almacenado en esa dirección no sea sobreescrito.

Pregunta 2

En lo siguientes ejercicios, deberá modificar la microarquitectura y/o la ISA del computador básico para que cumpla ciertos requerimientos. Todos estos cambios deben ser claramente especificados.

a) Modifique el computador básico, para que este utilice un esquema Von Neumann, *i.e.*, memoria de datos e instrucciones unificadas en una sola. **(3 ptos.)**
Solución: Desde el punto de vista de la microarquitectura, la idea principal para este ejercicio es dividir cada ciclo del clock en dos etapas. En la primera etapa, se incrementa el PC, que ahora está conectado a la memoria RAM compartida. Esta palabra (instrucción) es enviada a la unidad de control. Es importante que durante toda esta etapa, el resto del computador no sea alimentado con un flanco de subida. Una vez que la unidad de control el alimentada con la instrucción, comienza la segunda etapa, generando un flanco de subida para todos los elementos restantes, mientras que el PC no debe ser alimentado con un flanco de subida. De esta manera, la instrucción se ejecutará correctamente, terminando en el inicio de un nuevo ciclo, correspondiente al incremento del PC.
Desde el punto de vista de la ISA, se requiere que las instrucciones no permitan que los datos leidos de memoria lleguen a la unidad de control. El traspaso entre memoria y unidad de control solo debe ocurrir en la primera etapa del ciclo.

b) Modifique el computador básico para que este pueda detectar la posibilidad de que el stack sobreescriba datos almancenados en memoria. Agregue además la(s) instrucción(es) necesaria(s) para que un programa pueda manejar esta situación. **(3 ptos.)**
Solución: Para la microarquitectura, la idea es agregar **N** registros de 1 bit, donde **N** es la cantidad de palabras que puede almacenar la memoria. Estos registros indicarán si la palabra en esa posición ha sido escrita por una instrucción del tipo **MOV**, almacenando un 1. De esta manera, se asume que los registro parten con un 0 almacenado. Luego, se agrega un nuevo bit al registro status, **O**, cuyo valor se genera en base a la palabra apuntada actualmente por el registro **SP**, *i.e.* la salida del registro SP se conecta al bus selector de un MUX de **N** entradas, donde cada una de estas está conectada a uno de los registros de 1 bit. La salida de este MUX es el bit **O**.
Para que un programa pueda manejar esto, se agrega la instrucción JSO (**J**ump **S**tack **O**verwrite), que actua de manera similar al resto de las instrucciones de salto. En este caso, verifica si el bit **O** del registro status está en 1 para efectuar el salto.

Pregunta 3

a) Implemente, usando el *assembly* x86 y la convención *stdcall*, un programa que calcule el máximo común divisor de dos enteros no negativos, donde ambos no pueden ser 0 simultaneamente, utilizando el algoritmo de Euclides, descrito a continuación: **(3 ptos.)**

$$\text{Para } a, b \geq 0,\ mcd(a,b) = \begin{cases} a & \text{, si } b = 0 \\ mcd(b, a \bmod b) & \text{, en cualquier otro caso} \end{cases}$$

Solución:

```
        JMP    main
        a db    ?
        b db    ?
        res db 0
    main:
        MOV    AX, 0
        MOV    AL, b
        PUSH   AX
        MOV    AL, a
        PUSH   AX
        CALL   euclides
        MOV    res, AL
    euclides:
        PUSH   BP
        MOV    BP, SP
        MOV    BX, [BP-6]
        CMP    BL, 0
        JMP    setA
        MOV    AX, [BP-4]
        DIV    BL
        MOV    CX, 0
        MOV    CL, AH
        PUSH   CX
        PUSH   BX
        CALL   euclides
        JMP    return
    setA:
        MOV    AX, [BP-4]
    return:
        POP BP
        RET 4
```

b) Un computador utiliza una arquitectura Harvard + RISC, con las siguientes características principales:

- Sólo 3 registros, **A**, **X** e **Y**, todos de 8 bits, donde sólo el registro **A** tiene acceso a operaciones aritméticas o lógicas. Las únicas operaciónes aritméticas válidas para **X** e **Y** son INC X o INC Y y DEC X o DEC Y.
- Las operaciones aritméticas y lógicas de esta arquitectura, reciben sólo un parámetro, ya que afectan unicamente al registro **A**. Por ejemplo, ADD 0x10 sería igual a ADD A, 0x10. Las operaciones disponibles son las mismas que en x86, con excepción de MUL y DIV, que no existen en esta arquitectura.
- Múltiples instrucciones de carga, definidas de la siguiente manera:

Instrucción	**Descripción**	**Uso**
LD? #Lit	Carga en el reg. **?** el valor del literal **Lit**	LDA #0x01
LD? Dir	Carga en el reg. **?** el contenido de la dir. **Dir**	LDX 0x34
LDA Dir,?	Carga en el reg. **A** el contenido de la dir. dada por **Dir+?**	LDA 0x34,X
ST? Dir	Carga en la dir. **Dir** el valor almacenado en el reg. **?**	STA 0x7C
STA Dir,?	Carga en la dir. **Dir** el valor almacenado en la dir. dada por **A+?**	STA 0x7C,Y
TRANS Reg1,Reg2	Carga en el reg. **Reg1** el valor almacenado en el reg **Reg2**	TRANS A,X

Utilizando la arquitectura recién descrita, escriba un programa que calcule el producto punto entre dos vectores de números enteros no negativos. **(3 ptos.)**

3b)

Asumo que ambos arreglos/vectores son del mismo largo y que la operación no causa Overflow.

```
Length: db ? ;largo de los arreglos

Res: db ? ;resultado

Arr1: db ? ;puntero al inicio del arreglo 1

Arr2: db ? ;puntero al inicio del arreglo 2

Main:

      LDA #0x00 ;cargamos un 0 en A

      STA Res ;ponemos un 0 en el resultado

      LDX Length;cargamos un 0 en X

.loop

      TRANS A,X ;transferimos el valor de X a A

      CMP #0x00 ;comparamos si es igual al largo de los arreglos

      JGE .exit ;si es igual o mayor al largo salimos

      LDA Res

      PUSH A ;hacemos push del resultado

      CALL Mul ;multiplicamos [Arr1 + x] y [Arr2 + x] dejamos el
resultado de la multiplicación en Res

      POP A ;Recuperamos el valor del resultado en el registro A

      ADD Res ;sumamos el resultado de la multiplicacion con el resultado
actual

      STA Res ;cargamos el resultado a la variable Res

      DEX ;decrementamos X en 1 (DEC X)

      JMP .loop ;volvemos al loop

.exit

      ;resultado queda en Res
```

```
      RET
MUL:
      LDA Arr1,x ;pone el valor de Arr1 + x en A
      TRANS Y , A ;carga Arr1 + x en el registro Y
      LDA #0x00 ;carga un 0 en Res
      STA Res
.loop
      TRANS A , Y ;transfieree a Y al registro A
      CMP #0x00
      JEQ .exit ;si Y es 0 entonces retorna
      LDA Res ;carga el valor de Res en A
      ADD Arr2,x ;le suma Arr2,x a Res
      STA Res ;carga el valor en res
      DEY ;decrementa Y (DEC Y)
      JMP .loop ;vuelve al loop
.exit
      RET
```

1.5 por uso adecuado del ASM.

1.5 por el algoritmo

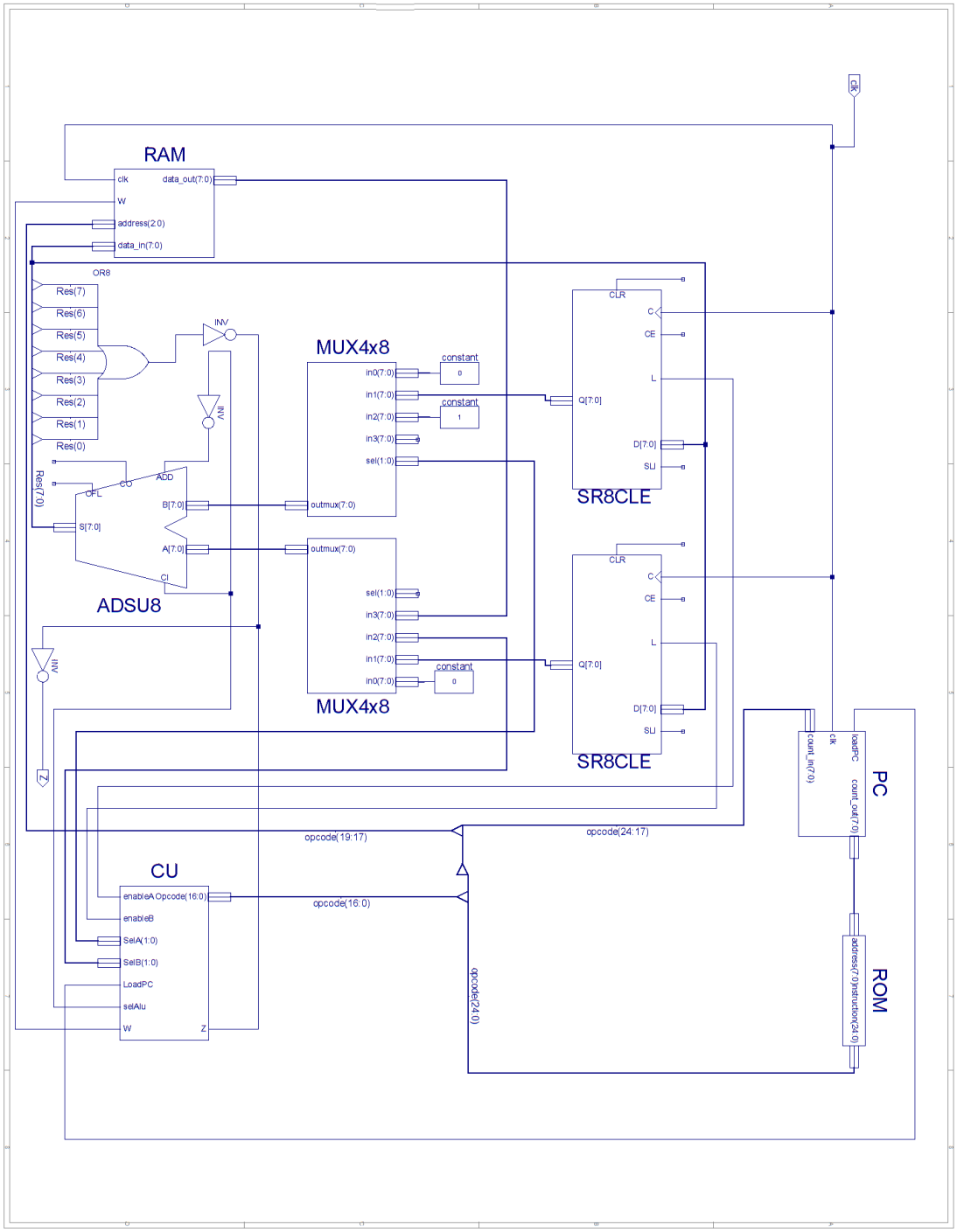
Si el uso del ASM tiene puntaje 0 la pregunta completa tiene 0.

Si la pregunta está incompleta recibe penalización en ambos puntajes.

Pregunta 4

a) La arquitectura es Harvard porque separa la memoria de instrucciones y la de datos.

b) Posible solución:



c) Posible solución, de acuerdo a la respuesta de b)

ADD A, B
“00000000 000000000 010 01 01 0”

SUB B, 10
“00001010 000000000 001 01 01 1”

INC B
“00000000 000000000 001 10 01 0”

MOV (3), A
“00000011 000000000 100 01 00 0”

MOV B, (7)
“00000111 000000000 001 00 01 0”