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

IIC2343 – Arquitectura de Computadores (II/2017)

Proyecto Semestral: Entrega Práctiva 04

Máquina programable con LCD externo, 50 % de la entrega 04

Fecha de entrega: Lunes 27 de Noviembre a las 12:00 horas

1. Descripción de la Evaluación:

Muy importante, leer por favor!

Para esta evaluación se va a hacer algo un poco diferente. Dado que es la entrega final del curso, queremos evaluar el avance que ha tenido cada persona de manera grupal e individual. Para lograr esto, tenemos dos opciones posibles que pueden elegir. Independientemente de cuál eligan, deben como grupo hacer la tarea 4 y la coevaluación.

1.1. Opción 1: Interrogación Escrita

Esta opción corresponde a una evaluación escrita sobre los contenidos del proyecto del curso. Es una evaluación individual y optar a ella significa renunciar a la entrega 4 que involucra a la placa. Esta decisión es personal. Si deciden esto, la nota de la parte física corresponderá a la nota de la evaluación escrita. Para expresar lo que desean hacer, deben rellenar este formulario: `https://goo.gl/forms/EspL50eNuZBIr0242` antes del **10 de noviembre**. El horario es el de la I3 del curso; la(s) sala(s) se avisará(n) más adelante.

Es importante mencionar que si tienen una nota menor a 4,0**, deben hacer la entrega 4.**

1.2. Opción 2: Entrega 4 + cierre de proyecto

Si se deciden por esta opción, antes de mostrar su entrega 4, deberán mostrarán como grupo el juego que deben desarrollar para esta entrega en la placa. El lugar de la entrega no está definido aún, pero pueden registrar un horario de disponibilidad para el día de la entrega acá: `https://goo.gl/sc5c9m`. Para pedir un horario deben dejar un comentario en la casilla correspondiente.

2. Objetivo (opción 2)

Para esta entrega tendrán que mejorar una vez más el computador básico desarrollado durante las entregas pasadas, extendiendo la cantidad de inputs y agregando la capacidad de output. Además deberán agregar soporte para un LCD externo y el manejo de todos los leds disponibles. Todo lo anterior para finalmente desarrollar un programa de un nivel superior con capacidad de interacción con usuarios.

En la Figura 1 se muestra el diagrama del computador básico modificado que tendrán que diseñar para esta entrega.

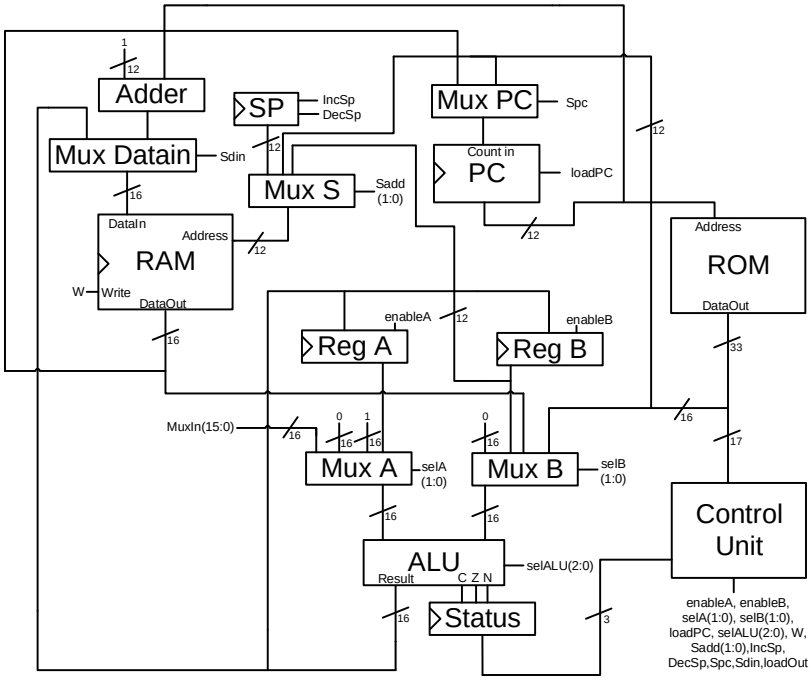

Figura 1: Computador básico.

Como se puede apreciar en la Figura 2, la implementación del *MuxIn* cambia un poco, incluyendo un componente *Timer* que entrega 3 enteros de 16 bits, con la información de segundos, milisegundos y nanosegundos. También se agrega un *Decoder OUT* y registros para almacenar la información a mostrar a través del display de 7 segmentos y leds de la placa.

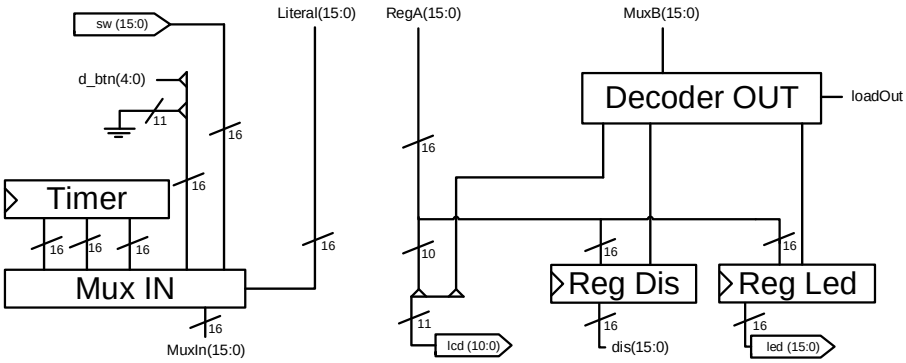

Figura 2: I/O detallado.

Para estandarizar el uso de los componentes, se reservarán los siguientes puertos para entradas y salidas del MuxIN:

Puerto	Input
0	Switches
1	Botones
2	Segundos
3	MSegundos
4	uSegundos
*	Nada

Figura 3: Tabla de puertos Input.

Puerto	Output
0	Display
1	Leds
2	LCD
*	Nada

Figura 4: Tabla de puertos Output.

3. Materiales

Para esta entrega se les entregarán los siguientes materiales:

1. Librería en assembly para el manejo de la pantalla LCD y el desarrollo del programa.
2. Manuales descriptivos del hardware para comprender el funcionamiento de la LCD y su librería.
3. Archivo *MegaMental.bit* que tiene un juego que ocupa I/O. Deben usarlo como ejemplo de una posible entrega y no copiarlo.

4. Desarrollo

Para desarrollar esta entrega se recomienda a los grupos realizar los siguientes pasos:

1. Conectar los LEDs restantes descomentando el .xdc y editando el bus correspondiente.
2. Agregar instancia de Timer (componente parte del proyecto base) y conectar sus salidas a MuxIN.
3. Agregar registros intermedios y conectar al Display y los LEDs.
4. Crear DecoderOUT y conectarlo a los componentes correspondientes.
5. Setear la señal speed del Clock_Divider a full ("00").
6. Desarrollar el programa editando la librería de acuerdo a lo necesario según los manuales (o sea, escribir su programa dentro del mismo archivo de la librería, de nombre 'Code - Base LCD').
7. Editar el .xcd para tener conexión con la LCD (revisar Pmod Header JB, Pmod Header JC) y añadir el puerto de salida adecuado a Basys3.

8. Creando el bus correspondiente a lo descomentado en el punto anterior. El bit de control que viene del DecoderOUT hacia el bus lcd (ver figura 2) corresponde al bit más significativo del bus. Los otros 10 son datos que llegan desde el registro A.
9. Leer manual de la LCD para comprender cómo comunicarse con la LCD.
10. Utilizar muxB para enviar valores a la LCD
11. Debuggear hasta que todo funcione.

5. Instrucciones y *assembler*

1. Al soporte de literales se debe agregar que puedan ser ingresados como caracteres, de la forma c.
2. Se debe poder definir caracteres como literal y arreglos de caracteres al principio de cada programa, los que deben ser de la forma:

```
1 DATA:
2 letrac 'c'
3 palabra "hola"
```

   Internamente, las frases se deben transformar en arreglos de caracteres terminados por un 0, por ejemplo [h, o, l, a, 0].

3. Manejar variables como punteros (dirección en memoria) y por su valor, tal que usar paréntesis indique que se quiere usar el valor en memoria de la variable y sin paréntesis indique el uso de su puntero:

```
1 DATA:
2 var 13
3
4 CODE:
5 MOV A, var    //mueve el puntero de var al registro A
6 MOV A, (var) //mueve el valor de var al registro A, o sea, 13
```

   Esto significa que el literal asociado a la instrucción, que es el mismo en ambos casos, debe tratarse como un MOV A,Lit en el primer caso (sin paréntesis) y en el segundo caso, se debe traer un valor desde la memoria. Esto se aplica a todas las instrucciones que utilicen valores en memoria y los guarden en un registro.

4. Intrucciones IN y OUT:

```
IN   A,Lit     (A = Input[Lit], usando la entrada Lit del MuxIN)
     B,Lit     (B = Input[Lit])
     (B),Lit   (Mem[B] = Input[Lit])

OUT A,B       (dirige A al puerto de valor B del Decoder OUT)
    A,(B)     (dirige A al puerto Mem[B])
    A,(Dir)   (dirige A al puerto Mem[Dir])
    A,Lit     (dirige A al puerto Lit)
```

   Las descripciones de los puertos está en la tabla dada más arriba.

6. Especificaciones de la funciones a programar

Para probar el computador hecho por cada grupo tendrán que generar un programa con interacción del usuario a través de los dispositivos de entrada y salida de la placa, usando el código en assembly base que se pondrá a su disposición. Este programa es un pequeño juego desarrollado por ustedes.

El juego que creen debe contar con lo siguiente:

1. El juego debe ser de 2 o más jugadores. El juego comienza pidiendo el nombre de ambos jugadores, donde cada uno puede ser de un máximo de 4 caracteres. El ingreso de estos nombres se debe hacer con los botones arriba, abajo, izquierda y derecha, y su confirmación se realiza con el botón central, mostrando el puntero solo durante la edición.

2. El juego debe rotar entre los jugadores para que ejecuten alguna acción. Cuando sea el turno de un jugador, deben mostrar su nombre en la línea superior.

3. Cada jugador debe tener un puntaje asociado que se modifique durante el juego. Se debe mostar este puntaje durante su turno junto a su nombre en la línea superior.

4. Al terminar una partida, se debe mostrar en pantalla el jugador ganador.

5. Se debe tener una opción de iniciar una nueva partida manteniendo los jugadores o de reiniciar el juego desde el comienzo.

6. Se debe usar la línea inferior de la pantalla para mostrar mensajes del juego.

7. Hacer uso creativo de los componentes de input (botones, switches) y output (LEDs, Display).

7. Ejemplos

Considere los siguientes ejemplos, los cuales corresponden a diversos programas que su *assembler* debe ser capaz de compilar:

- Programa 1:

```
DATA:

CODE:

MOV B,1          //   Dejar el Puerto de los Leds en el Registro B

MOV A,ABCDh      // |
OUT A,0          // | Mostrar "ABCD" en el Display

MOV A,AAAAh      // |
OUT A,B          // | Mostrar "1010101010101010" en los Leds

end:
JMP end          //   No Hacer Nada Ms
```

- Programa 2:

```
DATA:

CODE:            // Peudo Reloj | Velocidad de clock a "full"

loop:

 IN A,2          // |
 OUT A,0         // | Mostrar Segundos en el Display

 IN A,3          // |
 OUT A,1         // | Y Milisegundos en los Leds

JMP loop         //   Repetir
```

- Programa 3:

```
DATA:

odd 0            // Es impar
last 0           // Fue impar

CODE:            // Parpadeo Inverso | Velocidad de clock a "full"
```

```
7
8  loop:
9
10  IN A,2          // |
11  AND A,1         // |
12  MOV (odd),A     // | Segundo es Impar ?
13
14  IN B,0          //   Leer Switches
15
16  MOV A,0         // |
17  NOT A           // |
18  ADD A,(odd)     // |
19  XOR A,B         // | Si fue Par, Invertir Switches
20
21  OUT A,1         //   Resultado a Leds
22
23  MOV A,(last)    // |
24  CMP A,(odd)     // |
25  JEQ loop        // | Si Hubo Cambio
26
27   MOV B,1        // |
28   XOR (last)     // | Invierte Variable
29
30   IN A,1         // |
31   OUT A,0        // | Y Envia Botones al Display
32
33 JMP loop         //   Repetir
```

8. Entrega

Deben entregar:

- El proyecto completo de la CPU, es decir, todos los archivos involucrados en su proyecto en una carpeta con nombre entrega 4.
- Código del *assembler* y una explicacin de su funcionamiento.
- Un archivo de texto con el juego a programar escrito en *assembly*.
- Un breve informe (incluyendo el número de grupo y el nombre de los integrantes) que contenga la especificación de la estructura de las instrucciones de su CPU (función de cada uno de los 33 bits de una instrucción), más una tabla con todas las instrucciones soportadas por la CPU y su implementación de acuerdo a la especificación indicada anteriormente. Además, este informe debe contener la explicación de cómo ocupar su *assembler*, detallando paso a paso cómo usar su programa para ensamblar un archivo en *assembly* y generar el archivo de salida para insertar en la ROM. Si bien todos los *assembler* entregados debiesen ocuparse de la misma manera, de todas formas puede ser que existan pequeñas variaciones entre unos y otros, motivo por el cual deben explicar la forma de uso. Adicionalmente, este informe debe indicar específicamente qué hizo cada integrante del grupo durante la entrega (Un párrafo por persona).
- Dos breves párrafos en el informe que expliquen lo que fue más fácil y más difícil en la entrega.
- Explicación del funcionamiento de su juego.

La entrega del proyecto (código, informe y adicionales) es por medio del repositorio en Github tal que el Lunes 27 de Noviembre a las 12:00 horas deben tener en la rama Master todos los archivos correspondientes a su grupo. El día de la entrega deben llegar con sus archivos listos para mostrar a su ayudante asignado.

Entregas atrasadas serán **penalizadas con 0.5 puntos** por cada hora (o fracción) de atraso.

Para entrega 2 y 3: El día anterior a la entrega, se subirán los archivos para la evaluación tanto en la página del curso como en un issue en el Syllabus del curso. Son 6 archivos que corresponden a algoritmos en para evaluar. Deben generar los archivos .bit para cada uno de los ellos. Luego, durante la entrega presencial, se van a probar esos algoritmos y se pedirá la compilación completa (transformar el .txt a instrucciones de la ROM, insertar estas instrucciones en el computador y generar el bitstream en Vivado) de uno de ellos. Toda acotación especial se debe incluir en el informe de cada entrega o en un README.

9. Puntajes

A continuación se describe el puntaje asociado a cada item:

Elemento	Puntaje Asociado
Ingreso nombres de jugadores	0.4
Turnos	0.3
Mostrar puntajes	0.2
Uso componentes Input	0.4
Uso componentes Output	0.4
Mostrar Ganador	0.2
Nueva partida o nuevo juego	0.4
Uso LCD (muestra texto)	3
Fornalidad	0.3
Informe	0.4

Como podrán notar, el ítem con mayor puntos es el de la pantalla LCD. Para poder tener ese puntaje, debeán demostrar que lograron entender y ocupar la librería asociada a la LCD. En resumen, deben mostrar un programa hecho por ustedes que muestre texto tanto en la línea sueprior e inferior; el contenido de la pantalla debe cambiar a lo largo del programa (notar que esto significa que pueden optar a un 4 en la entrega en caso de que no logren terminar su juego).

10. Contacto

Cualquier pregunta sobre el proyecto, ya sean de enunciado, contenido o sobre aspectos administrativos deben comunicarse con los ayudantes creando issues en el Syllabus del Github del curso o directamente con los ayudantes:

- Francesca Lucchini: flucchini@uc.cl
- Felipe Pezoa: fipezoa@uc.cl
- Hernán Valdivieso: hfvaldivieso@uc.cl
- Luis Leiva: lileiva@uc.cl

11. Evaluación

Cada entrega del proyecto se evaluará de forma grupal y se ponderará por un porcentaje de coevaluación para calcular la nota de cada alumno.

Dado lo anterior, dentro de las primeras **24 horas** posteriores a cada entrega, **todos los alumnos** deberán completar de forma **individual y obligatoria** el formulario web que los ayudantes pondrán a su disposición, repartiendo un máximo de 4 puntos (aunque cambia según la cantidad de estudiantes en el grupo; el máximo corresponde al número de compañeros), con hasta un decimal, entre sus compañeros y una diferencia máxima entre el mayor y menor puntaje de 1. La suma de todos los puntos obtenidos por el integrante, *sp*, será utilizada para el cálculo de la nota de cada entrega, lo que puede hacer que este repruebe el curso.

La nota de cada entrega se calcula de la siguiente forma (con un máximo de 7,5):

$$NotaEntrega_{individual} = \text{mín}(k_g \times NotaEntrega_{grupal}, NotaEntrega_{grupal} + 0,5)$$

donde,

$$k_g = \frac{sp+3}{7}$$

y sp puede variar de acuerdo a la cantidad de estudiantes que conforman el grupo.

Los alumnos que no cumplan con enviar la coevaluación en el plazo asignado tendrán un **descuento de 0.5 puntos** en su nota de la entrega correspondiente.

12. Integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1,1 en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.