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

IIC2343 - Arquitectura de Computadores (II/2019)

Entrega 4

Entrega: Jueves 5 de Diciembre | 10:59:59 a.m.

Requisitos

- Esta entrega es de carácter grupal. Cualquier tipo de falta a la honestidad académica será sancionada con la **reprobación** del curso con la nota mínima.
- Los nombre de archivos y el cómo deben ser ejecutados son parte del formato, no respetarlo será penalizado.
- El programa de la **placa** deberá ser realizado en `VHDL`.
- La **documentación** deberá ser realizada en un archivo `Markdown` y subirlo junto a su tarea, de nombre README.md, en el mismo repositorio.
- Esta entrega deberá ser subida a su repositorio grupal de *GitHub* antes de la fecha y hora dada.

Objetivo

Para esta entrega tendrán que expandir el computador básico (una última vez), agregando la capacidad de input y output. Deberán agregar soporte para el manejo de todos los leds, *switchs* y botones disponibles. Todo lo anterior para finalmente desarrollar un programa de un nivel superior con capacidad de interacción con usuarios.

Adicionalmente, deberá estar completo su *assembler* con todas las instrucciones mencionadas en el enunciado de Assembler. **IMPORTANTE:** Debido a los acontecimientos a nivel nacional, el assembler **NO** debe aceptar caracteres de la tabla ASCII, ni tampoco definición de *strings*. Solo debe soportar las instrucciones.

Especificaciones CPU

Sin entrar todavía en detalles, el computador que deberán armar debe contar con:

- **Todos los componentes de la entrega anterior**.
- Un componente ***Timer*** que entrega la 3 enteros de 16 *bits*: segundos, milisegundos y microsegundos (se les entrega el componente en la carpeta de proyecto en el *syllabus*).
- Un **multiplexor IN**, el cual tiene un selector de 16 *bits*, para elegir las entradas, cada una también de 16 *bits*.
- Un **registro Dis** de 16 *bits* que guarde el valor del registro A, y que su output es conectado a los *displays* de 7 segmentos.
- Un **registro Led** de 16 *bits* que guarde el valor del registro A, y que su output es conectado a los *leds*.
- Un componente **Decoder Out** explicado más adelante.

En la Figura 1 se muestra el diagrama del computador básico recién descrito, donde se pueden apreciar cada uno de los bloques mencionados junto con las interconexiones pertinentes (además de la indicación del tamaño de cada bus). Cabe mencionar que todos los bloques síncronos de la Figura 1 son de flanco de subida.

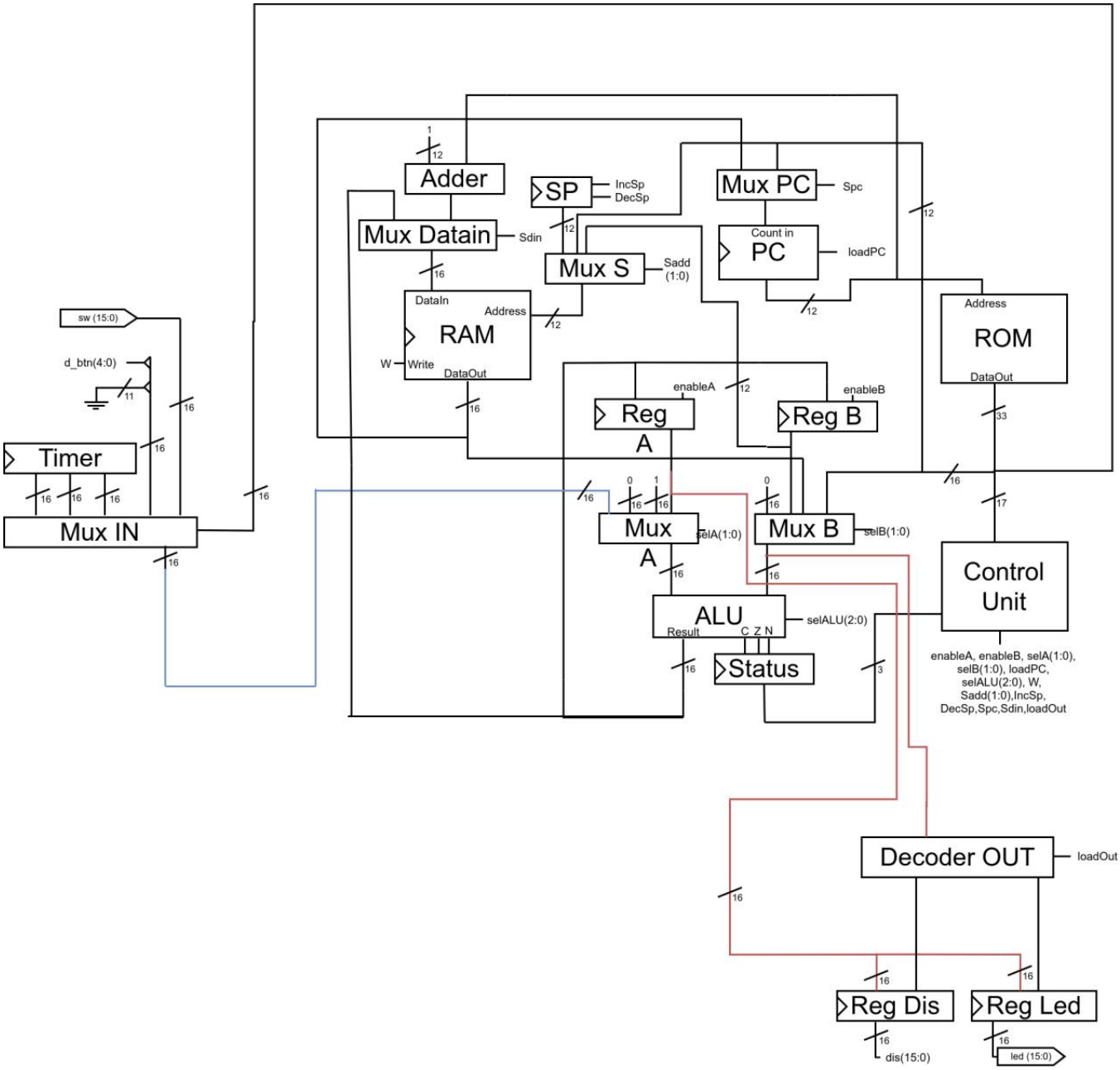

Figura 1: Computador básico.

Especificaciones del Input

Como se puede apreciar en la Figura 2, el contenido del bus de entrada esta determinado por el multiplexor `Mux IN`, que dependiendo de los 16 bits del literal selecionará una entrada distinta.

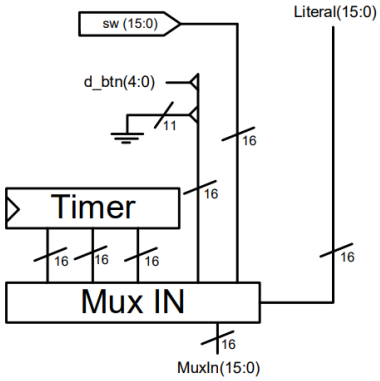

Figura 2: Input detallado.

Para estandarizar el uso de los componentes, se reservarán los siguientes puertos para entradas:

Puerto	Input
0	Switches
1	Botones
2	Segundos
3	mSegundos
4	μSegundos
*	Nada

Figura 3: Tabla de puertos Input.

- La señal `d_btn` corresponde a un vector con la información de la salida de instancias del componente `Debouncer`, que a su vez tiene de entradas señales del vector `btn`, correspondiente a la `Basys3` (no olvidar descomentarlo en los *constrains*). El orden de de las señales de `d_btn` deben ser las mismas que las de la señal `btn`. Estas son:
  - bit 0: central
  - bit 1: arriba
  - bit 2: izquierda
  - bit 3: derecha
  - bit 4: abajo.

  Esta señal será conectada como entrada al selector `Mux IN`, con el que el computador trabajará como uno de los medios de recibir señales de entradas humanas.

  Un ejemplo del uso de esta señal sería que si una persona presiona el botón izquierda y derecha, se activaría las instancias del componente `Debouncer` para dar salidas corregidas de ambos botones, y la señal `d_btn` sería igual a “00110” , mientras se tengan presionados dichos botones.

Especificaciones del Output

Como se puede apreciar en la Figura 4, se encuentra el componente `Decoder OUT` y registros para almacenar la información a mostrar a través del display de 7 segmentos y leds de la placa.

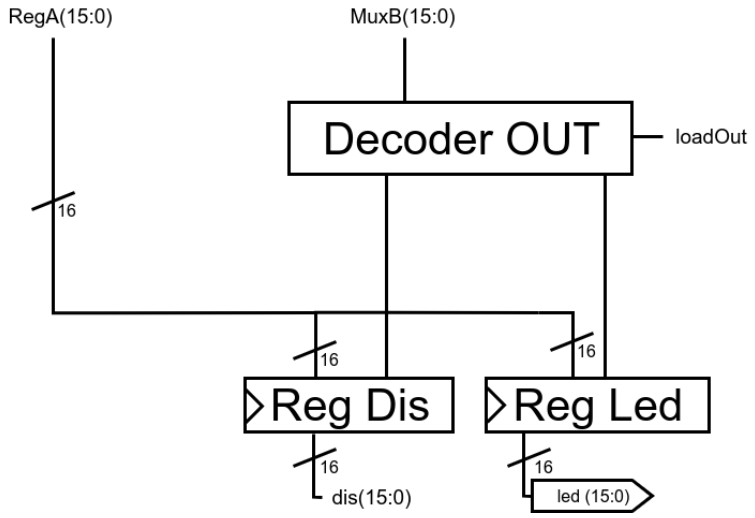

Figura 4: Output detallado.

Para estandarizar el uso de los componentes, se reservarán los siguientes puertos para salidas:

Puerto	Output
0	Display
1	Leds
*	Nada

Figura 5: Tabla de puertos Output.

- La señal `dis` corresponde a un vector de 16 bits con la información concatenada de las entradas `dis_a`, `dis_b`, `dis_c`, `dis_d` (ordenados respectivamente de más significativos a menos significativos). Dichas cuatro entradas son correspondientes al componente `Display_controller.vhd`

  Esta señal será conectada a la salida del registro *display* (`Reg Dis`), el cual a su vez tiene como entrada el valor del registro A, y una señal de carga dependiente del `Decorder OUT`(explicado más adelante). Con esto el computador tendrá una capacidad de output regulada por instrucciones a través del *display* de 7 segmentos.

  Un ejemplo de uso sería en la instrucción `OUT A,B` (con el selector B igual a cero). Al activar esta instrucción la señal dis pasaría a tener el valor del `Reg Dis`, que sería el valor en A.

- Ahora se explicará con más detalle el funcionamiento del `Decoder OUT`:

  El Decorder OUT corresponde a un Deccorder con enabler. La entrada de 16 *bits* de este componente se encargará de dirigir hacia donde se propaga la señal de `loadout`, proveniente de la Control Unit. Por otra parte, la señal `loadout` tiene 3 opciones donde ir a parar: la entrada de carga del `Reg Led`, la entrada de carga del `Reg Dis` o a ninguna parte. Esta señal será conectada a las entradas de carga de ambos registros (recordar que la señal de carga es la que permite actualizar los registros).

A continuación, la lista de instrucciones que su computador básico deberá soportar:

Entrega 2

MOV	A,B	guarda B en A
	B,A	guarda A en B
	A,Lit	guarda un literal en A
	B,Lit	guarda un literal en B
	A,(Dir)	guarda Mem[Dir] en A
	B,(Dir)	guarda Mem[Dir] en B
	(Dir),A	guarda A en Mem[Dir]
	(Dir),B	guarda B en Mem[Dir]
ADD SUB AND OR XOR	A,B	guarda A op B en A
	B,A	guarda A op B en B
	A,Lit	guarda A op literal en A
	B,Lit	guarda A op literal en B
	A,(Dir)	guarda A op Mem[Dir] en A
	B,(Dir)	guarda A op Mem[Dir] en B
	(Dir)	guarda A op B en Mem[Dir]
NOT SHL SHR	A	guarda op A en A
	B,A	guarda op A en B
	(Dir),A	guarda op A en Mem[Dir]
INC	A	incrementa A en una unidad
	B	incrementa B en una unidad
	(Dir)	incrementa Mem[Dir] en una unidad
DEC	A	decrementa A en una unidad
CMP	A,B	hace A-B
	A,Lit	hace A-Lit
	A,(Dir)	hace A-Mem[Dir]
JMP	Ins	carga Ins en PC
JEQ	Ins	carga Ins en PC si en el status Z = 1
JNE	Ins	carga Ins en PC si en el status Z = 0
NOP		no hace cambios

Entrega 3

MOV	A,(B)	guarda Mem[B] en A
	B,(B)	guarda Mem[B] en B
	(B),A	guarda A en Mem[B]
	(B),Lit	guarda Lit en Mem[B]
ADD SUB AND OR XOR	A,(B)	guarda A op Mem[B] en A
	B,(B)	guarda A op Mem[B] en B
NOT SHL SHR	(B),A	guarda op A en Mem[B]
INC	(B)	incrementa Mem[B] en una unidad
CMP	A,(B)	hace A-Mem[B]
JGT	Ins	carga Ins en PC si en el status N = 0 y Z = 0
JGE	Ins	carga Ins en PC si en el status N = 0
JLT	Ins	carga Ins en PC si en el status N = 1
JLE	Ins	carga Ins en PC Ins si en el status N = 1 o Z = 1
JCR	Ins	carga Ins en PC Ins si en el status C = 1
PUSH	A	guarda A en Mem[SP] y decrementa SP
	B	guarda B en Mem[SP] y decrementa SP
POP	A	incrementa SP y luego guarda Mem[SP] en A
	B	incrementa SP y luego guarda Mem[SP] en B
CALL	Ins	guarda PC+1 en Mem[SP], carga Ins en PC y decrementa SP
RET		incrementa SP y luego carga Mem[SP] en PC

Entrega 4

IN	A,Lit	guarda Input[Lit] en A
	B,Lit	guarda Input[Lit] en B
	(B),Lit	guarda Input[Lit] en Mem[B]
OUT	A,B	envia A a Output[B]
	A,(B)	envia A a Output[Mem[B]]
	A,(Dir)	envia A a Output[Mem[Dir]]
	A,Lit	envia A a Output[Lit]

Bitácora de proyecto

Adicionalmente, para esta entrega su grupo deberá actualizar su bitácora **en formato Markdown** que hable de los siguientes puntos:

- Explicación del funcionamiento de cada componente de su implementación del computador básico y con especial detalle en lo que se refiere la unidad de control (arquitectura interna y señales).
- La especificación de la estructura de las instrucciones de su CPU, es decir, la codificacíon de cada uno de los bits de su palabra de instrucción.
- El trabajo realizado por cada uno de los miembros de su grupo. Se debe indicar específicamente que hizo cada integrante del grupo y deben explicar que fue lo mas difícil para el grupo en la entrega.
- Especificaciones sobre el uso de su Assembler.
- Una tabla o referencia donde se indique que instrucciones son soportadas por su implementación de acuerdo a la especificación dada anteriormente.
- Especificación de reglamento del juego creado por el grupo (más detalles especificados abajo).

Requerimientos

- Soportar las instrucciones de la entrega pasada.
- Extender la arquitectura necesaria, de acuerdo al enunciado para soportar las nuevas instrucciones.
- Crea un ejecutable para escribir la ROM con un archivo Assembler.
- **Incluir el README.md, el Informe y los archivos correspondientes con lo solicitado.**
- Responder un **google form**, especificado más abajo.

Entrega

La entrega se realizará a través de GitHub. El repositorio debe contener una carpeta con su proyecto de Vivado, el ensamblador y los archivos `.bit`. En el caso de la carpeta del proyecto, deben subir solo la carpeta `.srcs`, el archivo `.xpr` y el archivo `Basys3.xdc` (las carpetas y archivos restantes se recomienda evitar su subida configurando el archivo `.gitignore`). Para el *assembler*, deberá subir su ejecutable o archivo `Python3` solicitado, el cual debe estar en su propia carpeta. Finalmente, los archivos `.bit` deben estar dentro de su propio directorio y tener el nombre correspondiente al algoritmo en assembly desde el que se creo. En ese mismo directorio deben estar los algoritmos compliados en formato `.srcs`. Deben entregar dos `.bit`, el primero corresponde a un juego especificado en este enunciado, y el segundo corresponde a un juego **creado por su grupo** (explicado más adelante).

En resumen, dentro de su repositorio deben tener tres carpetas, la del proyecto de Vivado, la del *assembler* y la de los archivos `.bit` junto con los `.asm`. Documentos como la bitácora y el `README`, debe estar en el repositorio, en el directorio raíz.

Evaluación

La evaluación de su entrega se enfocará en dos partes principales:

I.- Especificaciones de programa regular

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. El programa que deben crear es un juego llamado **Adivina la palabra** y el flujo de este se describe en los siguientes puntos:

1. El programa inicia mostrando el número de su grupo por 3 segundos en el Display 7 segmentos, luego el display vuelve a 0 dando inicio al juego.
2. El juego inicia con el jugador 1 escribiendo una palabra binaria en los switches. Luego, presiona el botón UP para confirmar la palabra escrita. Finalmente, desordena los switches para confundir al otro jugador
3. El jugador 2 recibe la placa con 8 leds encendidos, indicando las veces que puede intentar adivinar.
4. Al escribir una palabra en los switches y presionar el boton UP, pueden ocurrir dos cosas:
   - La palabra es incorrecta, por lo que uno de los leds se apaga, indicando que un intento fue utilizado. El display 7 segmentos, muestra la cantidad de bits en los que te equivocaste. (Distancia de Hamming)
   - La palabra es correcta, haciendo que los leds en posición par se enciendan, y cada un segundo, estos se apagan y encienden los de posición impar, y así sucesivamente (luces de navidad).
5. Al quedarse sin intentos (todos los leds apagados) el display 7 segmentos muestra el numero (1057)

II.- Especificaciones de programa grupal

Este programa es un juego como el anterior, pero aquí se busca que **sean creativos** y que apliquen todos los mecanismos de input (botones y *switchs*) y **todos** los mecanismos de output (*display* de siete segmentos y *leds*).

Como juego debe tener una forma de ganar y perder además que debe tener reglas **claras y concisas**, las cuales deben ser enviadas antes del **27 de Noviembre a las 11:59 a.m.**, a través del siguiente link.

Estas reglas pasarán por una revisión de ayudantes para determinar si cumple con los requerimientos mencionados anteriormente, junto con un criterio para evitar realizar un programa trivial o uno excesivo. De no cumplir con la revisión, los ayudantes **rechazarán su propuesta**, y deberán realizar una versión modificada por el cuerpo docente.

Con esta sección se busca que dejen llevarse con su creatividad y apliquen lo aprendido durante el curso para realizar algo entretenido, por lo que los invitamos a **ser creativos y divertirse!**

III.- Regulaciones mínimas

- El uso correcto de Vivado y escritura en VHDL, es decir, que los archivos se desplieguen correctamente al abrir el archivo .xpr y que estos sean capaces de compilar el .bit con ambas ROM.
- El correcto funcionamiento de su ensamblador para la escritura de la ROM.
- El correcto funcionamiento de los archivos .bit solicitados.
- El contenido de la bitácora. Este debe explicar con claridad los elementos que se pidieron.
- **Evaluación de Pares.** Deberán contestar un form semanalmente respecto al trabajo que han realizado juntos. Deberán indicar si han trabajado como grupo y quienes han trabajado. No contestar este form significará un descuento en su nota. **Lo anterior depende de la situación país**.

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.