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

IIC2343 – Arquitectura de Computadores (I/2020)

Proyecto Semestral: Entrega Práctica 04

Computador básico con input y output

Fecha de entrega: Viernes 3 de Julio a las 20:00 horas

1. Objetivo

Para esta entrega tendrán que mejorar una vez más el computador básico desarrollado durante las entregas pasadas. Esto es agregando los componentes *Reg Dis*, *Reg Led* y *Decoder OUT*

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

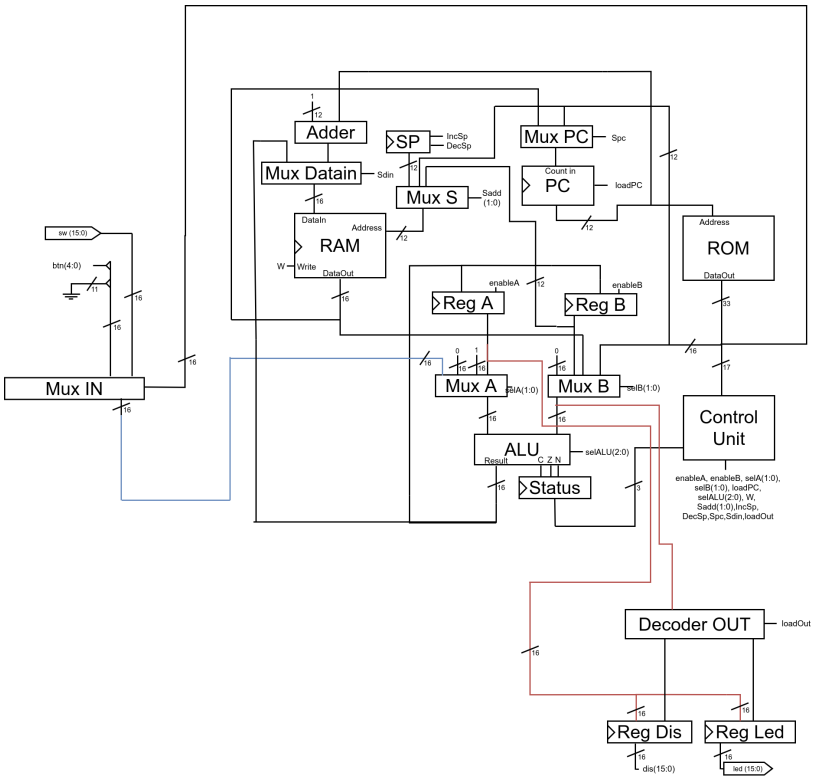


Figura 1: Computador básico con input y output

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 y leds.

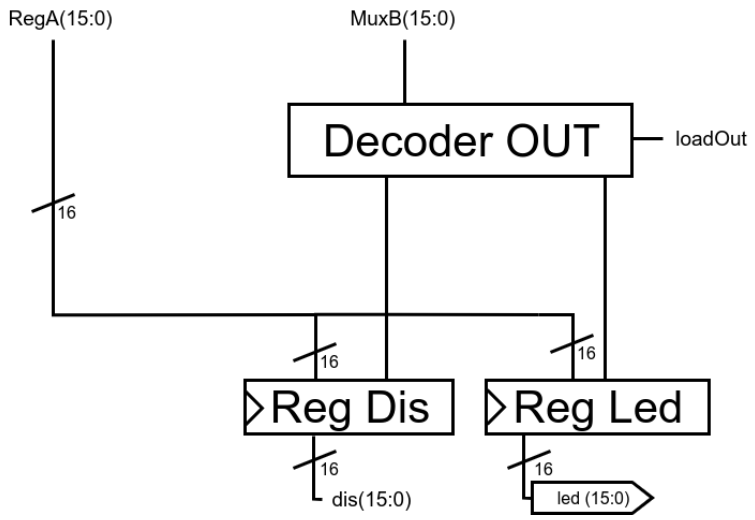


Figura 2: Output detallado.

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

Puerto	Output
0	Display
1	Leds
*	Nada

Figura 3: 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).

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 Decorder 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).

2. Instrucciones

A continuación las nuevas instrucciones que deberá soportar su computador:

```
OUT A,B     (envía A a Output[B])
    A,(B)   (envía A a Output[Mem[B]])
    A,(Dir) (envía A a Output[Mem[Dir]])
    A,Lit   (envía A a Output[Lit)
```

Evaluación

La evaluación de su entrega se enfocará en el desarrollo de un computador funcional que tenga cargado dos programas en forma de subrutina, y al cual se pueda ser seleccionado y al cual se le pueda pasar parámetros:

I.- Especificaciones de ejecución del un programa

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 debe indicar cuando esta listo para recibir inputs (luego de haber cargado toda la DATA), activando todas las leds (la signals led debe ser igual a FFFF en la simulación). En el programa debe ser capaz de recibir el input del botón izquierdo para ejecutar el programa 1, o el botón derecho para ejecutar el programa 2. Una vez ingresado el programa se sigue los siguientes pasos:

1. Inicialmente el display muestra el valor 1010 (en hexadecimal), indicando que se esta recibiendo los parámetros.
2. Se recibe los parámetros necesarios con los switches, se apretará el botón abajo indicar que ese será el parámetro utilizado en la subrutina.
3. Cuando se reciba los parámetros necesarios para ejecutar la subrutina se apretará el botón central que inicie el programa. **Nota:** puede suponer que siempre se dará los parámetros correctos.
4. Cuando inicia un programa las leds se apagan indicando que se esta en ejecución.
5. Cuando el programa termina el resultado es mostrado en el display.
6. Finalmente se encienden las leds indicando que el programa termino.

II.- Elección de los programas:

Cada programa debe estar escrito como una subrutina (es decir debe ser accesida por CALL y RET). Cada subrutina debe ser entregada en un archivo code-programX.txt y ambos programas deben ser contenidos en un programa principal llamado code-ROM.txt que tiene la lógica de inputs y output descrita en la primera sección.

Se puede escoger dos de los cuatro programas en esta lista (solo se evaluará dos, si ingresa más de dos no será considerado):

- Subrutina exponente: recibe primero la base, luego recibe el exponente y da como resultado la operación exponente de esa base.

- Subrutina división entera: recibe primero el dividendo, luego el divisor y da como resultado el cociente.
- Subrutina factorial: recibe primero un entero, luego da como resultado el factorial de ese número.
- Subrutina ASCII: recibe primero un entero codificado en ASCII (un número entero del 48 al 57), luego recibe un segundo número ASCII, luego recibe un tercer número ASCII, y da como resultado los tres números ordenados en el display de menor a mayor (desde el más significativo al menos significativo).

3. Desarrollo

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

1. **Tener completada la entrega 3:** Partir de la base que se tiene todos los requerimientos de dicha entrega para poder avanzar correctamente en el desarrollo de esta entrega. En caso de que falte algún componente o instrucción no testeada, el grupo debe asegurarse de completar y testear el computador de la entrega pasada.
2. **Componentes:** Crear e importar los componentes nuevos del diagrama del computador en el archivo Basys3, procurando conectar correctamente las señales de entradas y salidas.
3. **Opcode y Señales:** Analizar y plantear si la forma en como la *ROM* entrega las instrucciones a la *Control Unit* es suficiente para agregar las nuevas instrucciones. Se recomienda ampliar las tablas que identifiquen el código de operación y sus respectivas señales asociadas.
4. **Comportamiento de los Componentes:** Completar los comportamientos específicos de cada uno de los componentes a desarrollar. Estos son: *Registro Display*, *Registro Led*, *DecoderOUT*.

   a) *Registro Display*: Un registro de 16 bits que contiene como entrada la salida de registro A y su señal de carga esta definida por el Decoder OUT.

   b) *Registro Led*: Un registro de 16 bits que contiene como entrada la salida de registro A y su señal de carga esta definida por el Decoder OUT.

   c) *Decoder OUT*: 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).
5. **Desarrollar los programas:** Desarrollar un código para los dos programas escogidos. **Recordar que debe seguir estructura de subrutina**, si su programa solo contiene saltos tendrá descuento.
6. **Simulación:** Finalmente, teniendo todo lo anterior listo, deberán simular y observar si los valores transitivos y finales de los display corresponden a los valores esperados o no.

4. Requerimientos

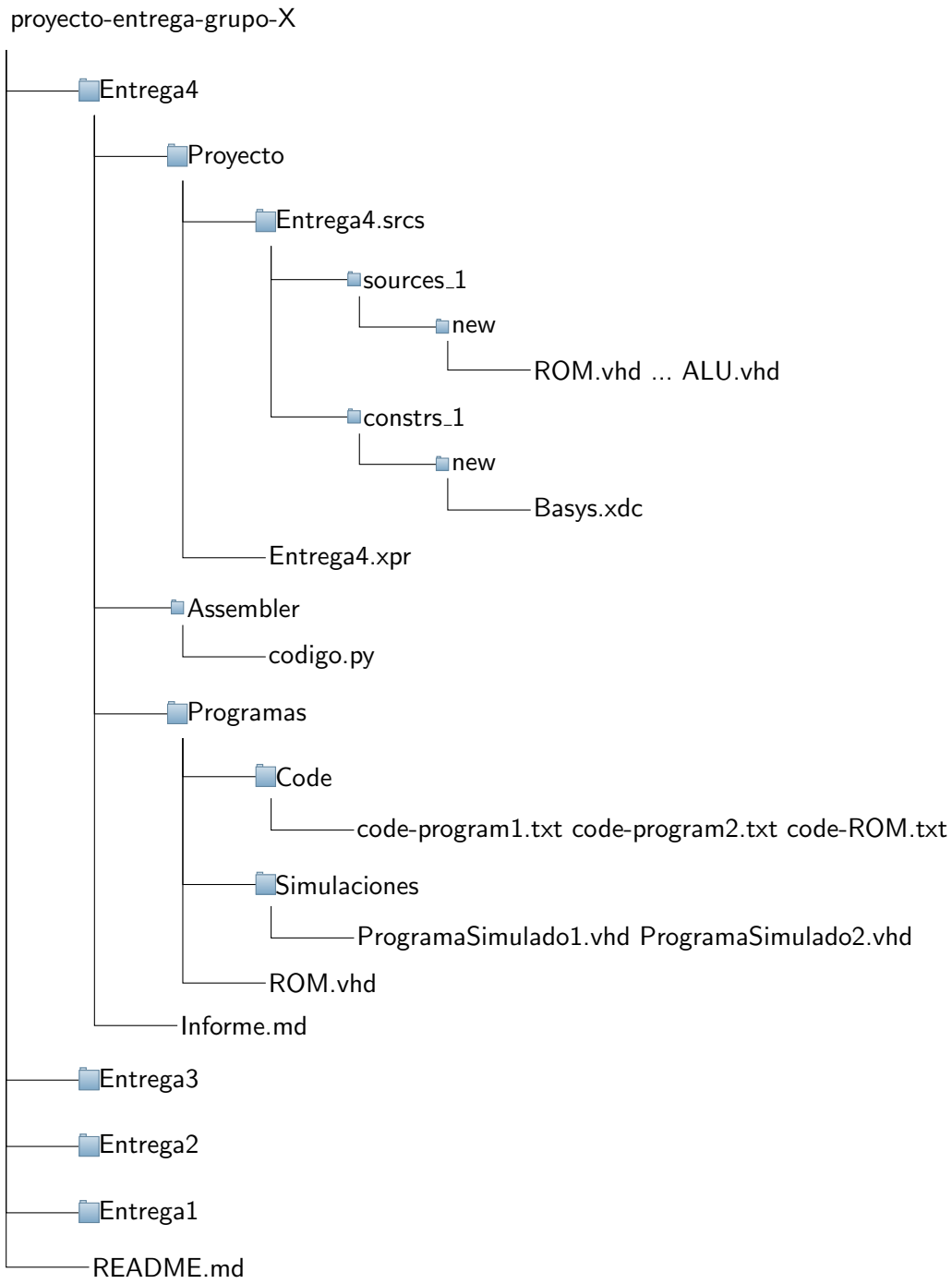
A continuación se describen los requerimientos de esta entrega y el puntaje asociado a cada item:

- Para implementar declaraciones condicionales **solamente** se permite hacer uso de bloques `with/select`. El uso de los *statements process*, `case` e `if/else` quedan absolutamente prohibidos. Esto porque se privilegia el uso de selectores y operaciones lógicas básicas para el desarrollo del proyecto. **Para esta entrega, queda estrictamente prohibido utilizar cualquier tipo de librería aritmética que simplifique la tarea de la suma**. ***Nota:*** Esto solo aplica a los componentes a desarrollar, en los archivos de simulación se permite el uso de process.
- Incluir el archivo .xpr de su proyecto en su entrega (no incluirlo implicará un descuento por no cumplir con lo pedido).
- (6 pts) Se evaluara el contenido dentro de la entidad `Basys3.vhd`
  - (3 pts) Programas correctos, bien visualizados y bien simulados en la Basys3:
    - (1 pts) Correcto diseño del algoritmo y ejecución del programa 1
    - (1 pts) Correcto diseño del algoritmo y ejecución del programa 2
    - (1 pts) Correcto diseño del algoritmo y ejecución del programa principal
  - (1 pts) Correcta arquitectura de la CPU.
  - (1 pts) Correcta implementación de la *Control Unit* con las tablas de opcode y señales consistentes.
  - (1 pts) Contenido del informe.
  - Puede crear más *sources* para facilitar el problema.

Incluir su informe en formato Markdown : En este debe un archivo llamado `Informe.md` que describa 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.

Además, se deberá incluir resultados de los archivos de simulación, junto con los archivos ROM.vhd asociados, mostrando que su entidad efectivamente resuelve el problema. Para esto basta con mostrar el valor del output del Registro display en las señales de los display y el output del Registro Led em las señales delas leds.

Formato del repositorio: El formato del respositorio debe ser el siguiente:



NOTA: La carpeta Assembler es opcional, solo en caso de que hagan un programa para ello. Si traspasan los test a código de máquina manualmente no se debe subir, y lo deben especificar en su informe. **EL NO SEGUIR ESTE FORMATO DE REPOSITORIO PUEDE SIGNIFICAR UN DESCUENTO EN SU CALIFICACIÓN FINAL.**

5. Entrega

Deben entregar:

- La carpeta con su proyecto de Vivado. En el caso de la carpeta del proyecto, deben subir solo la carpeta `.srcs`, el archivo `.xpr` y el archivo `Basys3.xdc` dentro de la carpeta new de los constrains (ver el diagrama de más arriba).
- Una carpeta Programa que contenga dos carpetas. Una carpeta de *Simulaciones* con sus archivos de simulaciones para cada test y una carpeta *Code* con las subrutinas de los programas escogidos, y el programa principal que las contiene. Adicionalmente la ROM de este programa principal debe estar en este mismo directorio.
- Su archivo *informe.md*

La entrega del proyecto (código, informe y adicionales) es por medio del repositorio en Github tal que el día 03 de Julio a las 20 horas deben tener en la rama Master todos los archivos correspondientes a su grupo.

6. Evaluación de pares

La evaluación de pares estará activa durante la entrega y hasta el miércoles 01 de Julio a las 23 horas. El link es el siguiente:

`https://forms.gle/f3VsPSaAUoukXHnm8`.

Esto con el objetivo que nos informen si algún integrante de su grupo no se comunica o no se compromete con el proyecto para que, como equipo docente, poder intervenir antes de la entrega y encontrar soluciones posibles para el correcto desarrollo de la misma.

7. Assembly

Esta es la lista de instrucciones separadas por entrega.

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
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
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]
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
IN	A,Lit	guarda Input[Lit] en A
	B,Lit	guarda Input[Lit] en B
	(B),Lit	guarda Input[Lit] en Mem[B]

Entrega 4

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]

8. Contacto

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

- Felipe Valenzuela: frvalenzuela@uc.cl
- Matías López: milopez8@uc.cl
- Cristóbal Herreros ceherreros@uc.cl
- Raúl Del Río Jara rjdelrio@uc.cl

9. 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.