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

IIC2343 – Arquitectura de Computadores (II/2017)

Proyecto Semestral: Entrega Prctica 02

Computador básico y compilador assembly, 70 % de la entrega 02

Fecha de entrega: Lunes 25 de Septiembre a las 17:00 horas

1. Objetivo

Para esta entrega tendrán que diseñar y armar su propio computador básico **de 16 bits**, el cual utilizarán posteriormente para ejecutar programas hechos en lenguaje assembly. Lo que deberán armar en Vivado es su propia CPU, la cual deberá cumplir con una serie de requisitos descritos más abajo, dentro de los cuales se encuentra soportar una lista determinada de instrucciones en assembler.

Paralelamente, tendrán que crear su propia conversión de assembler a binario, la cual estará dada por la estructura que escojan para las palabras de instrucción. Además, deberán programar su propio *assembler* que soporte las instrucciones y especificaciones detalladas en el documento "Assembler", que está disponible en el Syllabus del curso.

2. Especificaciones CPU

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

- Dos registros de 16 bits (registros A y B).
- Una memoria ROM de instrucciones de 4096 palabras de 33 bits cada una.
- Una memoria RAM de 4096 palabras de 16 bits cada una. Esta memoria es de lectura asíncrona y escritura síncrona.
- Una unidad aritmético lógica (ALU) capaz de realizar ocho operaciones (todas las de la Entrega 1 menos la multiplicación).
- Un *program counter* (PC) de 12 bits de direccionamiento.
- Dos multiplexores de cuatro entradas de 16 bits.
- Una unidad de control de lógica combinacional.
- Un registro de status (C, Z, N).

En la Figura **??** 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 porte de cada bus). Cabe mencionar que todos los bloques síncronos de la Figura **??** son de flanco de subida.

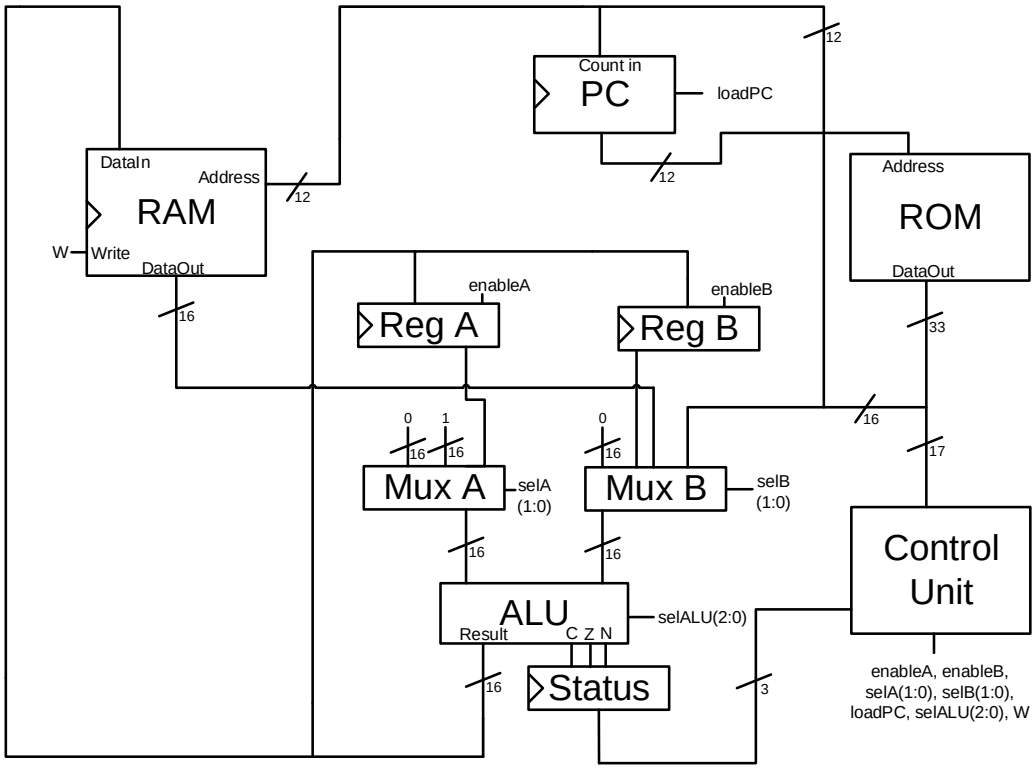

Figura 1: Computador básico.

3. Ejemplos

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

- Programa 1:

```
1 DATA:
2 CODE:          // Canten 'La Farolera':
3 MOV A,2        // 2
4 MOV B,2        // y 2
5 ADD A,B        // son 4
6 NOP            // 4
7 NOP            // y 2
8 NOP            // son
```

```
ADD A,2          // 6
NOP              //
NOP              // 6
NOP              // y 2
ADD A,B          // son 8
MOV B,A          // y 8
ADD A,B          // 16
NOP              //
NOP              //
NOP              // A = 10h , B =8h
```

- Programa 2:

```
DATA:

CODE:            // Swaps

MOV A,3          // A = 3
MOV B,5          // B = 5

MOV (0),A        // |
MOV A,B          // |
MOV B,(0)        // | Swap con MOV y variable auxiliar

SUB A,B          // A = 2

XOR A,B          // |
XOR B,A          // |
XOR A,B          // | Swap con XOR
```

- Programa 3:

```
DATA:            // Variables a sumar

 a 5
 b Ah

CODE:            // Sumar variables

MOV A,0          // 0 a A
ADD A,(a)        // A + a a A
ADD A,(b)        // A + b a A
MOV B,A          // Resultado a B

end:
 DEC A           // A--
JMP end
```

- Programa 4:

```
DATA:

 a E5h          // 11100101b
 b B3h          // 10110011b
 bits 0b

CODE:           // Contar bits en 1 compartidos

MOV A, ( a)     // a a A
AND A , ( 1d )  // A & b a A
JMP loop        // Empieza desde loop

bit:
 INC  (2h)      // bits ++
 loop:
  CMP A ,0b     // Si A == 0
  JEQ  end      // Terminar
  SHR A         // Si A >> 1 genera carry
  JCR  bit      // Siguiente desde bit
                // Si no
  JMP  loop     // Siguiente desde loop

end:
 MOV A,(10b)    // Resultado a A
 JMP end
```

- Programa 5:

```
DATA:

 varA 8
 varB 3

CODE:           // Restar sin SUB ni ADD:

MOV A,(varB)    // varB a A
NOT (varB),A    // A Negado a varB
INC (varB)      // Incrementar varB

suma:

MOV A,(varA)    // Resultado a A

end:
 NOP
 JMP end
```

- Programa 6:

```
DATA:

CODE:           // No debe saltar

JMP start

error:
 MOV A,FFh      // FFh a A
 JMP error

start:
 MOV B,1
 MOV A,B
 INC A
 CMP A,B
 JEQ error

 INC B
 CMP A,2
 JNE error

 MOV (0),A
 INC B
 CMP A,2
 JGT error
 CMP A,(0)
 JGT error

 INC B
 INC (0)
 CMP A,(0)
 JGE error

 INC B
 CMP A,2
 JLT error

 CMP A,1
 JLT error
 INC B
 DEC A
 CMP A,0
 JLE error

 INC B
 SHL A
 JCR error
```

```
48
49  SUB A,3
50  JCR error
51
52  MOV A,11h       // 11h a A
```

4. Puntajes

A continuación se describe el puntaje asociado a cada item:

La entrega consistirá en mostrar el output de diferentes algoritmos en en el display. El display, al igual que en la entrega anterior, deberá mostrar los 8 bits menos significativos de cada registro, pero ya no debe mostrar el resultado de la ALU. Entonces, se evaluará que la secuencia mostrada en los displays sea correcta. Si es correcta, se obtiene el puntaje completo, en caso contrario no se obtiene puntaje.

Cada ítem de la tabla está asociado a un ponderador. En la entrega cada ítem se pude evaluar entre 0 y 1 y ese valor se multiplcia con el ponderador asociado. En el caso del descuento y bonus, se puede evaluar entre 0 y 0,5.

1	0.5	0.5	0.5	0.5	1	0.5	0.5	1	1
Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Formalidad	Informe	Bonus	Descuento

Figura 2: Tabla de puntajes.

5. Entrega

Deben entregar:

- El proyecto completo de la CPU, es decir, todos los archivos involucrados en su proyecto, sin las carpetas .caché, .hw, .runs o .sim (eso debe ser manejado a través del .gitignore).
- Una carpeta para la tarea (Assembler y Disassembler).
- Un breve informe (con el número de grupo y el nombre de los integrantes en portada) que contenga
  1. La especificación de la estructura de las instrucciones de su CPU (función de cada uno de los 33 bits de una instrucción).
  2. Tabla o referencia a una tabla con todas las instrucciones soportadas por la CPU y su implementación de acuerdo a la especificación indicada anteriormente.
  3. 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.
  4. Este informe debe indicar específicamente qué hizo cada integrante del grupo durante la entrega (Un párrafo por persona) y una sección que explique qué fue lo más fácil y lo más difícil para el grupo en la entrega.

La entrega del proyecto (código, informe y adicionales) es por medio del repositorio en Github tal que el Lunes 25 de Septiembre a las 17:00 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.

6. 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

7. 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.

8. 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.