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

IIC2343 - Arquitectura de Computadores (II/2024)

Etapa 1 del Proyecto

Desarrollo y comunicación

Durante el desarrollo del proyecto, se les asignará a su grupo un repositorio en GitHub, donde deberán subir cada una de las etapas que van realizando. Las dudas **generales** del proyecto deberán ser realizadas mediante *issues* en el repositorio del curso, mientras que para las dudas más específicas (como temas de código) deberán agendar una pequeña reunión online con alguno de los ayudantes de laboratorio o realizar issues en sus repositorios privados. Adicionalmente, podrán resolver las dudas que tengan durante las instancias de laboratorio antes de cada entrega.

Para cualquier otro caso, **deberán** contactar a la coordinadora de proyecto: Catalina Miranda (`ccmiranda1@uc.cl`).

Descripción

Su trabajo en esta primera etapa será exclusivamente dentro de la CPU y la ROM.

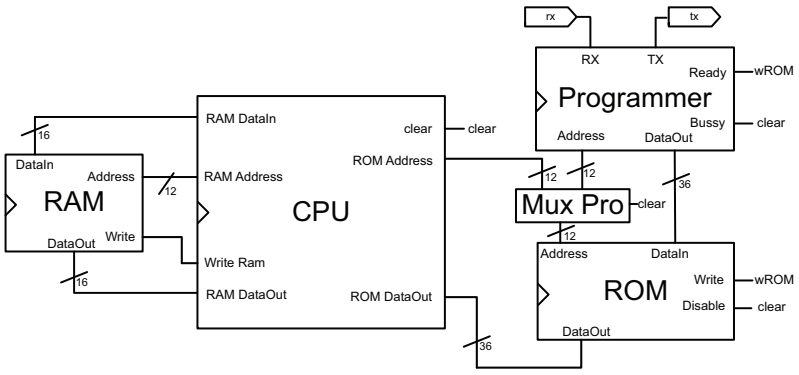

Figura 1: Diagrama parcial del computador básico de proyecto dentro del componente Basys3.

Comenzarán el proyecto implementando las características que le permitan hacer operaciones de carga, aritmético-lógicas y saltos en su computador. Para esto, tendrán que implementar la arquitectura descrita en el diagrama.

Los desafíos de esta etapa corresponden al diseño de su unidad de control y a la creación de su *assembler*. El primero implica diseñar el lenguaje de máquina que se almacenará en las palabras de **36 bits** de

la ROM. Para ello, tendrán que decidir la distribución del literal y el *opcode* dentro de la palabra; y elegir la codificación de los bits dentro del *opcode* para que su unidad de control pueda, **solamente a través de lógica combinacional**, determinar las distintas señales de control que deben ser enviadas a cada componente. Finalmente, deberá conectar los registros a los *display* dentro de su CPU para poder visualizar los valores que se encuentran en ellos.

El segundo desafío corresponde a realizar el primer acercamiento a su *assembler*. Toda la información sobre su desarrollo se encuentra en el enunciado del *assembler* ya publicado en el módulo de Proyecto en Canvas.

Finalmente, deberán escribir un programa usando solamente las instrucciones de *assembly* especificadas en esta entrega, pero no es necesario que pueda ser ejecutado en su computador.

Hardware

A diferencia del computador básico visto en clase, los registros, ALU y palabras de la memoria de datos (RAM) son de 16 bits, las direcciones de la RAM y la ROM son de 12 bits y la ALU no tiene soporte para operar sobre números en complemento de 2, es decir, tanto los operandos como el resultado se interpretan como números positivos en todo momento.

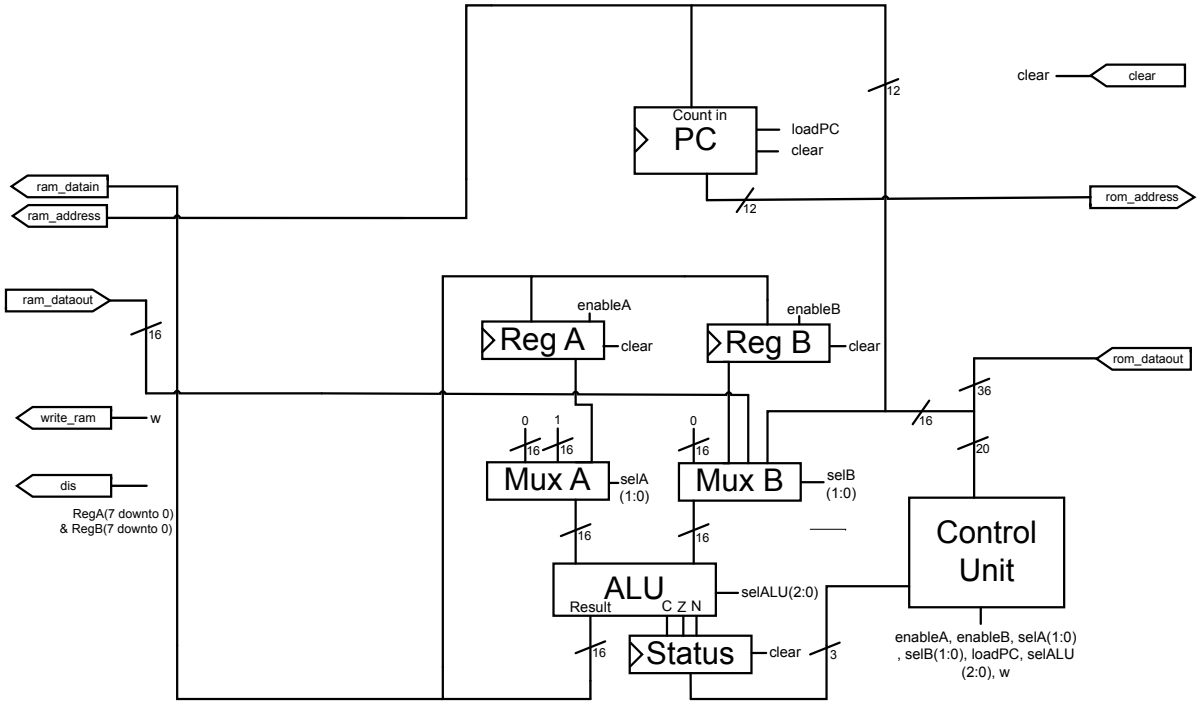

Figura 2: Diagrama interno de la CPU reducida del computador básico de proyecto.

Componentes

Su proyecto **debe** tener lo siguiente:

- Dos **registros** `Reg A` y `Reg B` de uso general, de 16 *bits* cada uno.
- Una **unidad aritmética lógica** `ALU` con 8 operaciones de 16 *bits*.
- Un **registro contador** `PC` para el *address* de la instrucción del programa, de al menos 12 *bits*.
- Dos **multiplexores** `Mux A` y `Mux B`, cada uno de cuatro entradas de 16 *bits*.
- Una **unidad de control** `Control Unit` implementada con **lógica combinacional**.
- Un **registro** `Status` para las *flags* `C`, `Z` y `N`.

** **Hint**: No es necesario crear diferentes componentes para cada una de ellas, recuerden que en vivado pueden crear distintas **instancias** de una componente para los elementos que tienen el mismo comportamiento. **

Assembler

Para evitar tener que traducir manualmente las instrucciones de *assembly* a código de maquina (1's y 0's) y escribirlas en la ROM, es necesario tener un programa que pueda realizar esta tarea y que pueda automatizar la escritura a la ROM. Este programa es conocido como *assembler* y, para esta entrega, es necesario comenzar a desarrollarlo.
Todas las instrucciones para desarrollar su *assembler* se encuentran en el enunciado de este mismo, que tiene especificado lo que debe hacerse para esta entrega y lo que se debe dejar pendiente para la siguiente. También tendrán a su disposición la siguiente página. Con esta podrán testear su el *output* de su *assembler* y la arquitectura de su computador de manera independiente.

Software

Además de realizar el *hardware* del computador, deben escribir un programa en *assembly* **para uno** de los siguientes problemas (que será asignado al azar para su grupo):

1. Determinante de matrices 2x2.
2. Separar un número entre centenas, decenas y unidades.
3. Palíndromo binario.
4. Factorial de un número.
5. Secuencia de sumatoria alternante.

Los datos necesarios para resolver cada uno de estos problemas se encontrarán en la sección DATA, por ejemplo, si le tocara resolver el tercer problema, el número a identificar si es palíndromo se encontraría guardado en DATA. Además, el *assembly* utilizado para escribir este código debe ser el implementado en esta etapa del proyecto, es decir, solo pueden usar instrucciones que se encuentren en la tabla de instrucciones.
El formato de entrega de este programa es **un archivo .txt** por grupo. **No se evaluarán** códigos que estén en algún otro formato como .pdf, .md, etc. Además, el código debe incluir una sección DATA en donde se definirán las variables necesarias y una sección CODE, en donde se encontrará el código realizado para resolver el problema elegido. Pueden ocupar todas las variables que consideren convenientes.
Finalmente, el nombre de su archivo debe tener el siguiente formato:

`Problema_<nro_problema>_Grupo_<nro_grupo>.txt`

Es decir, si un alumno fuera del grupo 34 y le tocara el problema 3 a resolver, su archivo debería llamarse

`Problema_3_Grupo_34.txt`

Archivos que no cumplan este formato no serán corregidos, y no recibirán puntaje en el apartado de Software.

Presentación

Se aceptarán envíos de *commits* hasta **el lunes 30 de septiembre a las 14:30 horas**, momento en que se recolectarán todos los repositorios. Deberán presentar sus proyectos **en los computadores del laboratorio en no más de 15 minutos** en el día y horario correspondiente a su sección. Esto se hará con una descarga del comprimido de su repositorio, que el equipo docente proveerá al momento de evaluar. El día de la presentación deberán mostrar **presencialmente** su proyecto y correr los test proporcionados por el equipo docente. Además, deberán mostrar y explicar brevemente el algoritmo implementado para resolver el problema que les fue asignado.

A la hora de la entrega, su repositorio **deberá contener todos los archivos necesarios para correr su proyecto**, es decir:

- Archivos de Vivado necesarios para su proyecto
- Archivo .txt del algoritmo
- Código de su assembler

Assembly

Etapa 1

MOV	A,B	Guarda el valor de B en A
	B,A	Guarda el valor de A en B
	A,Lit	Guarda un literal Lit en A
	B,Lit	Guarda un literal Lit en B
	A,(Dir)	Guarda el valor de Mem[Dir] en A
	B,(Dir)	Guarda el valor de Mem[Dir] en B
	(Dir),A	Guarda el valor de A en Mem[Dir]
	(Dir),B	Guarda el valor de B en Mem[Dir]
ADD SUB AND OR XOR	A,B	Guarda el resultado de A op B[1] en A
	B,A	Guarda el resultado de A op B en B
	A,Lit	Guarda el resultado de A op Lit en A
	B,Lit	Guarda el resultado de A op Lit en B
	A,(Dir)	Guarda el resultado de A op Mem[Dir] en A
	B,(Dir)	Guarda el resultado de A op Mem[Dir] en B
	(Dir)	Guarda el resultado de A op B en Mem[Dir]
NOT SHL SHR	A	Guarda el resultado de op A en A
	B,A	Guarda el resultado de op A en B
	(Dir),A	Guarda el resultado de op A en Mem[Dir]
INC	A	Incrementa el valor de A en una unidad
	B	Incrementa el valor de B en una unidad
	(Dir)	Incrementa el valor de Mem[Dir] en una unidad
DEC	A	Decrementa el valor de A en una unidad
CMP[2]	A,B	Ejecuta la instrucción SUB A,B sin actualizar el valor de A
	A,Lit	Ejecuta la instrucción SUB A,Lit sin actualizar el valor de A
	A,(Dir)	Ejecuta la instrucción SUB A,(Dir) sin actualizar el valor de A
JMP	Ins	Carga la dirección de la instrucción Ins en PC
JEQ	Ins	Carga la dirección de la instrucción Ins en PC si en Status se cumple Z = 1
JNE	Ins	Carga la dirección de la instrucción Ins en PC si en Status se cumple Z = 0
JGT	Ins	Carga la dirección de la instrucción Ins en PC si en Status se cumple N = 0 y Z = 0
JGE	Ins	Carga la dirección de la instrucción Ins en PC si en Status se cumple N = 0
JLT	Ins	Carga la dirección de la instrucción Ins en PC si en Status se cumple N = 1
JLE	Ins	Carga la dirección de la instrucción Ins en PC si en Status se cumple N = 1 o Z = 1
JCR	Ins	Carga la dirección de la instrucción Ins en PC si en Status se cumple C = 1
NOP[3]		No hace cambios

[1] En este caso, op representa la operación realizada: ADD, SUB, AND, OR o XOR.

[2] Notar que si bien no actualiza A, sí actualiza el registro Status.

[3] Es decir, el computador ejecuta una iteración sin realizar cambio alguno.

Puntaje

Para esta entrega, su nota final consistirá en una ponderación entre lo logrado en el hardware y el software. La nota de la parte de hardware (N_H) será evaluada mediante un *test* de correctitud que evalúa cada instrucción pedida en la entrega, el puntaje obtenido en este apartado dependerá de cuantas instrucciones correctas se lograron implementar segun el *test*. Mientras que la nota de la parte de software (N_S) dependerá de la correctitud de su algoritmo.

De esta manera, su nota final de la entrega será calculada como:

$$NE_1 = N_H * 0{,}7 + N_S * 0{,}3$$

Cabe mencionar que, para evaluar la parte de hardware, es necesario que **su *assembler* pueda procesar todas las instrucciones**.

Es de suma importancia mencionar que cualquier proyecto en el que se use *process* en alguna componente que no lo incluya originalmente, **obtendrán el puntaje mínimo en la entrega.** En otras palabras, queda totalmente prohibido su uso.

Recuperación de puntaje y requisitos

Durante las semanas previas a la presentación de la entrega, se realizarán salas de ayuda en el horario de laboratorio para que trabajen en el proyecto y resuelvan sus dudas con sus ayudantes. Si su grupo termina con un 100 % de asistencia en estas instancias, se le permitirá **recuperar hasta la mitad del puntaje descontado en la evaluación**. Para que la asistencia sea considerada, **al menos una persona** del grupo debe estar presente **durante los dos bloques de laboratorio** que correspondan a su sección. **Solo se aceptará asistencia al laboratorio de su sección**, es decir, cualquier persona que intente asistir a un laboratorio que no le corresponde no será admitido y su asistencia no será registrada.

La recuperación del puntaje, en caso de cumplir con los criterios de asistencia, se realizará durante la **próxima** sala de ayuda que corresponda a su sección (semana del 7 de octubre). En esta, debe mostrar que se logró arreglar los errores de su entrega original. El código de la recuperación deberá ser presentado el día de su laboratorio correspondiente. Este deberá incluir los arreglos de esta entrega y **nada posterior a ella**.