Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2343 - Arquitectura de Computadores

Tarea 6

29 de junio de 2019

Entrega: Martes 2 de Julio 20:00hr

1^{er} semestre 2019 - Yadran Eterovic

Requisitos

- Esta tarea es estrictamente individual y el curso, incluyendo alumnos, ayudantes y profesor, se encuentran afecto al código de honestidad académica.
- El nombre del archivo es parte del formato, no respetarlo será penalizado con la nota mínima.
- Todas las preguntas deben ser respondidas en base al material del curso, apoyadas con cálculos y ejemplos. Agregue todo lo que considere necesario con tal de obtener una respuesta completa.
- El uso de material externo debe acompañarse con argumentos que apoyen su uso y debe ser citado según corresponda. No es necesario seguir un formato en específico. Con que quede clara la referencia y su origen sea comprobable, es suficiente.
- Cualquier material sin referencia y el uso de tareas pasadas será sancionado con la nota mínima y citación con el profesor a cargo.
- Recordar que las tareas están para que aprendan, no hay beneficio en buscar atajos.
- Esta tarea deberá ser subida a su repositorio personal de *GitHub* correspondiente en la fecha y hora dada.

Pregunta 1

Se les presenta una simplificación de un computador básico capacitado para operaciones sobre 4 registros, acceso a memoria, saltos y operaciones Aritméticas lógicas básicas. Es su deber:

- Identificar las señales de control necesarias para el correcto funcionamiento del computador básica.
- Señalar en una tabla de verdad, el *opcode* del computador más las señales de control junto a los valores que deben tomar para cada instrucción.
- Diseñar, con ayuda de esta tabla, un *Instruction decoder* que decodifique las instrucciones de la *Instruction Memory* para ejecutar la ISA adjunta `en su totalidad`. Solo puede utilizar multiplexores y las compuertas básicas vistas en clases para construirlo. Si quiere hacer uso de componentes más específicos puede diagramarlos como una caja negra y diseñarlos a parte.

Hint: Aunque no es obligatorio, puede resultar más simple haciendo uso de Mapas de *Karnaugh*.

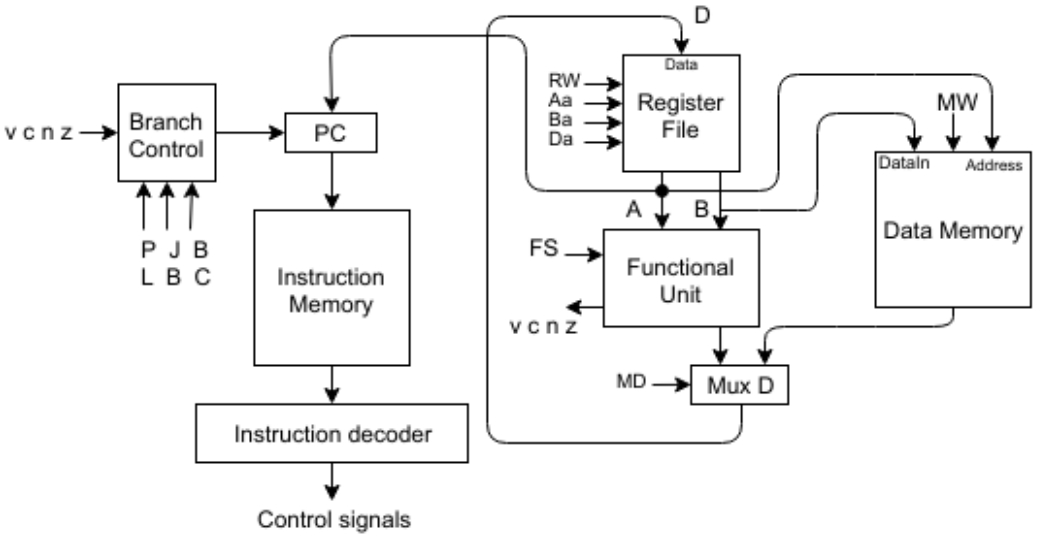


Diagrama Computador[1]

El *Program Counter* es controlado por el *Branch Control*, el cómo funciona dependerá netamente de las señales de control. No necesitan preocuparse de su estructura ni de cómo reconoce las condiciones necesarias para su ejecución. La siguiente tabla simplifica su uso

[1]Las conexiones entre las líneas se pueden indicar con puntos. Los punto son una indicación inequívoca de que las líneas están conectadas. Si la conexión es obvia, tipo `T`, no en necesario usar un punto.En un diagrama, Para más información sobre terminología en circuitos pueden ver este Link.

según las señales correspondientes[2]. Toda operación que utiliza la *Functional Unit* hace que el PC avance con normalidad.

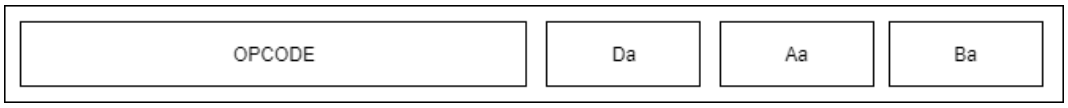
Operación PC	**PL**	**JB**	**BC**
Count Up	0	X	X
Jump	1	1	X
Branch on Negative	1	0	1
Branch on Zero	1	0	0

La *Functional Unit* (`FU`) solo está capacitada para ejecutar operaciones aritmética-logicas con los valores de entrada `A` y `B` que vienen del *Register File*. Su ISA asociada se presenta a continuación. Sus *Opcodes* dependen netamente de la señal FS.

Operación	**Señales de control (`FS`)**
`FU = A`	`000`
`FU = A + B`	`001`
`FU = A + ¬ B`	`010`
`FU = A ∧ B`	`011`
`FU = A ∨ B`	`100`
`FU = A ⊕ B`	`101`
`FU = ¬ A`	`110`
`FU = B`	`111`

El *Register File* contiene 4 registros, los cuales son seleccionados a modificar a través de la señal de destino D-Address (Da). Este se modifica según la señal Register-Write (RW) y los buses de salida A y B contienen el data almacenado en los registros elegidos según las señales A-Address (Aa) y B-Address (Ba).

Las instrucciones que se guardan en la *Instruction Memory* siguen la concatenación entre `Opcode` y los bits que seleccionan los registros de Destino (`D`), `A` y `B`. El siguiente diagrama intentará explicar el formato de cada una de las instrucciones del componente:



[2]Una `X` significa un *Don't care*, donde su valor puede ser 0 ó 1 sin influir en el resultado

ISA computador

Instrucción	Operación	Opcode
MOVA	R[Da] = R[Aa]	0000
ADD	R[Da] = R[Aa] + R[Ba]	0001
SUB	R[Da] = R[Aa] - R[Ba]	0010
AND	R[Da] = R[Aa] $\wedge$ R[Ba]	0011
OR	R[Da] = R[Aa] $\vee$ R[Ba]	0100
XOR	R[Da] = R[Aa] $\oplus$ R[Ba]	0101
NOT	R[Da] = $\neg$ R[Aa]	0110
MOVB	R[Da] = R[Ba]	0111
LD	R[Da] = M[Aa]	1000
ST	M[Aa] = R[Ba]	1001
BRZ	if (R[Aa] == 0)	1010
BRN	if (R[Aa] $>$ 0)	1011
JMP	PC = R[Aa]	1100

Pregunta 2

1. Haciendo uso de la micro-arquitectura anterior, modifique el computador para que pueda ejecutar las siguientes instrucciones:

Instrucción	Operación
INC	R[Da] = R[Aa] + 1
DEC	R[Da] = R[Aa] - 1
SHR	R[Da] = sr(R[Ba])
SHL	R[Da] = rl(R[Ba])
LDI	R[Da] = M[Lit]
ADI	R[Da] = R[Aa] + Lit

   Si su implementación genera algún cambio con respecto al funcionamiento ó micro-arquitectura del computador, debe indicarlo y explicar qué se debe hacer para que no cambie la ejecución del resto de instrucciones. Además, para cada instrucción, debe indicar las señales a activar (sean del diagrama original o las añadidas por usted) para su correcta ejecución. Puede realizar un diagrama o explicar en detalle su implementación.

2. Asumiendo que ya posee el computador de la implementación anterior, ¿qué otra instrucción podría incluir? No es necesario que mencione las señales involucradas para implementarla o el *Opcode* asociado, pero sí que justifique por qué se podría implementar.

Pregunta 3

1. Explique cómo funciona la transferencia de direcciones y datos desde/hacia los dispositivos *mapeados* a memoria.
2. ¿Con qué tipo de dispositivos de **I/O** es preferible utilizar *mapeo* de memoria sobre puertos?
3. ¿Por qué es mejor hacer uso de interrupciones en vez de *polling*? Mencione un ejemplo.
4. ¿Cuál es la función del vector de interrupciones? ¿Cuál es su contenido?
5. Explique la diferencia entre las interrupciones realizadas por *hardware* y *software*, dando un ejemplo de cada una.

Pregunta 4

Un robot simple, conectado a un computador, es accesible mediante *mapeo* a memoria. Este robot se mueve en una grilla infinita, donde cada celda puede estar vacía o contener una muralla. El robot tiene comandos para ser prendido, apagado, avanzar 1 espacio hacia adelante, girar a la izquierda en 90°, girar a la derecha 45°, examinar lo que hay adelante y voltear 180°. Cada vez que el robot se encuentra desocupado, *i.e.* ha sido recién iniciado o ha terminado una acción, genera una interrupción para informar que es posible darle un nuevo comando.

1. Describa el mapa de memoria necesario para manejar el robot.
2. Defina el formato de los datos que recibirá el robot como comandos y que entregará este para informar su estado.
3. Escriba en *Assembly* x86 la ISR asociada al control del robot, siguiendo el siguiente comportamiento: El robot avanza hasta encontrar una muralla, en cuyo caso girará a la derecha hasta encontrar un espacio vacío para avanzar, teniendo la precaución de que el robot no retroceda. Asuma que el espacio a sido diseñado para que el robot no se quede pegado girando eternamente.

Pregunta 5

¿Qué problema podría existir en un sistema operativo con soporte para memoria virtual, si **muchos procesos** se encuentran en ejecución **simultánea**? En caso de existir uno, indique cómo solucionarlo desde el punto de vista del uso de memoria, detallando claramente los elementos y estructuras involucradas. En caso de no existir problema, justifique el por qué.

Entrega y evaluación

La tarea se debe realizar de **manera individual** y la entrega se realizará a través de un cuestionario en el Siding. El formato de entrega debe consistir en un archivo llamado `tarea6_xxxxxxxx.pdf`, donde `xxxxxxxx` debe reemplazarse por su número de alumno.

El formato de creación es libre, incluso se pueden hacer a mano. En caso que las tareas sean hechas a mano, deben ser fotografiadas y transformadas a `PDF`. No se corregirán tareas que no sean legibles.

Se considerará una bonificación de 5 décimas si realizan su tarea en LaTeX. De hacerlo, debe subir un `.rar/.zip`, siguiendo el mismo formato del nombre a su repositorio, junto al PDF y todos los elementos necesarios para compilar el `.tex`

Archivos que no compilen y/o que no cumplan con el formato de entrega implicarán nota **1.0** en la tarea, sin excepciones. En caso de atraso, se aplicará un descuento de **1.0** punto por cada 6 horas o fracción.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes 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. 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 (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), 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 (grupo) copia un trabajo, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por

otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.