Práctica 4 - Diseño de ISA

Organización del Computador 1

Cuatrimestre Verano 2019

Ejercicio 1 Dada una arquitectura con palabras de *32 bits*; diga cuántos *bits* son necesarios para especificar una dirección de memoria en los siguientes casos:

a) Tamaño de memoria física: *4GB*; con direccionamiento a *byte*.

b) Tamaño de memoria física: *8GB*; con direccionamiento a *"medias palabras"*.

c) Tamaño de memoria física: *16GB*; con direccionamiento a *palabra*.

d) Tamaño de memoria física: *32GB*; con direccionamiento a *"palabra doble"*.

Ejercicio 2 Dada una arquitectura con palabras e instrucciones de y *bytes* que trabaja con una memoria física de x *bytes*, con direccionamiento a *palabra*:

a) ¿Cuántos *bits* serán necesarios para especificar una dirección cualquiera de la memoria?

b) ¿Cuál sería el número máximo de códigos de operación posibles suponiendo que todas las instrucciones incluyen sólo un operando con modo de direccionamiento directo a memoria?

c) ¿Cómo reescribiría las respuestas a las dos preguntas anteriores si x y y fuesen potencias de dos (i.e., $x = 2^k$ y $y = 2^j$)?

d) ¿Bajo que condiciones de j se obtiene la máxima granularidad de acceso a la memoria?

Ejercicio 3 ¿Cuál es el máximo número de instrucciones de una dirección que admtirían cada una de las siguientes máquinas?

a) Instrucciones de *12 bits*, direcciones de *4 bits*, y *6* instrucciones de dos direcciones.

b) Instrucciones de *16 bits*, direcciones de *6 bits* y n instrucciones de dos direcciones.

Ejercicio 4 Dada una máquina con instrucciones de *16 bits* y direcciones de *4 bits*, diseñe un formato de instrucción que contenga:

I. *15* instrucciones de *3* direcciones;

II. *14* instrucciones de *2* direcciones;

III. *31* instrucciones de *1* dirección;

IV. *16* instrucciones sin direcciones.

Ejercicio 5 Diseñe un formato de instrucción con código de operación extensible (la cantidad de *bits* del código de operacion es variable) que se pueda codificar en una instrucción de *36 bits* y permita lo siguiente:

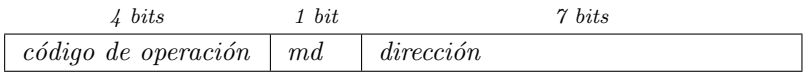
- *7* instrucciones con dos direcciones de *15 bits* y un número de registro de *3 bits*;

- *500* instrucciones con una dirección de *15 bits* y un número de registro de *3 bits*;
- *50* instrucciones sin direcciones ni registros.

Ejercicio 6 Suponiendo que se necesitan *3 bits* para direccionar un registro, ¿es posible diseñar un formato de instrucción cuyo código de operación sea extensible y permita codificar lo siguiente en una instrucción de *12 bits*?

- *4* instrucciones con tres registros;
- *255* instrucciones con un registro;
- *16* instrucciones sin registros.

Ejercicio 7 Sea un procesador que cuenta con un registro distinguido, llamado *acumulador*, sobre el que se realizan operaciones aritméticas; por ejemplo, sumar al acumulador un dato de memoria. El formato de las instrucciones que ejecuta es el siguiente:



Mediante el valor del bit *md* se escoge el modo de direccionamiento a utilizar: 1 para *directo* y 0 para *indirecto*.

Suponiendo que el código de la operación de suma mencionada más arriba es 1100, y el estado de la computadora es el indicado en la tabla, ¿Cuál será el contenido del acumulador después de ejecutar las siguientes instrucciones?

a) 1100 1011 0100

b) 1100 0011 0101

Acompañe el resultado con una explicación gráfica.

Memoria	
Dirección	Contenido
0x0034	0033h
0x0035	0036h
0x0036	0035h
0x0037	0034h

Registros	
Nombre	Contenido
Acumulador	2B58h

Ejercicio 8 Dados cuatro modelos de computadoras con direcciones de *16 bits*, códigos de operación de *8 bits* y tamaño de instrucciones múltiplo de *4 bits*; pero que pueden diferenciarse por las siguientes características:

Máquina 0: es una máquina *de pila*; sus instrucciones no precisan operandos (exceptuando PUSH y POP).

Máquina 0	
PUSH [M]	push [M]
POP [M]	[M] ← pop
ADD	push(pop+pop)
SUB	push(pop-pop)
MUL	push(pop*pop)
DIV	push(pop/pop)

Máquina 1: utiliza un registro *acumulador* (A) e instrucciones con un operando.

Máquina 1	
LOAD [M]	A ← [M]
STORE [M]	[M] ← A
ADD [M]	A ← A + [M]
SUB [M]	A ← A - [M]
MUL [M]	A ← A * [M]
DIV [M]	A ← A / [M]

La máquina 2 y la máquina 3 disponen de *16* registros e instrucciones de dos y tres operandos, que operan con cualquier combinación de direcciones de memoria y registros.

Máquina 2	
MOV X, Y	X ← Y
ADD X, Y	X ← X + Y
SUB X, Y	X ← X - Y
MUL X, Y	X ← X * Y
DIV X, Y	X ← X/Y

Máquina 3	
MOV X, Y	X ← Y
ADD X, Y, Z	X ← Y + Z
SUB X, Y, Z	X ← Y - Z
MUL X, Y, Z	X ← Y * Z
DIV X, Y, Z	X ← Y/Z

a) Escriba, para cada máquina, el programa que computa $x = (a + b * c)/(d - e * f)$.

b) ¿Cuántas instrucciones tiene cada programa?

c) Diseñe el formato de instrucción para cada uno.

d) ¿Cuál es la longitud (en *bytes*) del programa para cada computadora?

e) ¿Cuál es la cantidad de accesos a memoria en cada caso?

Ejercicio 9 Sea una arquitectura que posee *1024* registros de *64 bits*. Cada registro se divide en cuatro partes de *16 bits* que pueden ser referenciadas independientemente. La memoria física es de *2 GB* direccionables a *byte*.

El procesador provee el repertorio de instrucciones listado a continuación; estas instrucciones se describen pensando a cada registro Ri como un vector de cuatro posiciones, referenciadas como Ri(0), Ri(1), Ri(2) y Ri(3).

CLR R$u(i)$	Pone en *0* la componente i de u.
INT R$u(i)$, Rv, Rw	Almacena en la componente i de u el producto escalar entre v y w.
VEC Ru, Rv, Rw	Almacena en u el producto vectorial entre v y w.
ADD Ru, Rv, Rw	Almacena en u la suma componente a componente entre v y w.
SDV Ru, Rv, R$w(i)$	Almacena en u la división de cada componente de v por la componente i de w.
MOV p, q	Copia *64 bits*; p y q pueden ser registros completos o direcciones de memoria (todas las combinaciones son válidas).

a) Diseñe un formato de instrucción de longitud fija de *64 bits*.

b) ¿Cuáles modos de direccionamiento emplea el formato de instrucción propuesto?

c) ¿Qué longitud de palabra resultaría más adecuada para ese formato?

Ejercicios tipo parcial

Ejercicio 10 El procesador AiChip utiliza palabras de *8 bits* y direcciones de *8 bits*. Implementa un direccionamiento *a palabra* y la aritmética del procesador es complemento a dos. Cuenta con 4 registros de propósito general (R_0 a R_3) de 8 bits de longitud cada uno.

Provee el siguiente conjunto de instrucciones:

Instrucción	Efectos	Instrucción	Efectos
`ADD Ri Rj`	$R_i \leftarrow R_i + R_j$	`SWAP Ri Rj`	$Ri \leftrightarrow Rj$
`JG Ri Rj`	$if\ R_i > R_j\ then\ PC \leftarrow R_0$	`LOAD Ri`	$R_0 \leftarrow [R_i]$
`STORE Ri`	$[R_i] \leftarrow R_0$	`NOT Ri`	$Ri \leftarrow not(R_i)$
`ZERO Ri`	$R_i \leftarrow 0$	`LCTEL cte4`	$R_0\{0 \dots 3\} \leftarrow cte4$
`LCTEH cte4`	$R_0\{4 \dots 7\} \leftarrow cte4$		

Aclaraciones: $i, j \in [0, 3]$. `cte4` es una constante de *4 bits*. `LCTEH` carga `cte4` en los *bits* más significativos de `R0`, mientras que `LCTEL` carga la constante de 4 bits en los menos significativos.

a) ¿Cuál es el tamaño máximo de memoria que podrá utilizar este procesador?

b) ¿Cuál debe ser el tamaño (en bits) del stack pointer? ¿Por qué?

c) Diseñar un formato de instrucción de **longitud fija** de una palabra para el AiChip.

d) Es posible agregar instrucciones sin operandos. ¿Cuántas? Indicar la codificación de las mismas.

e) Sin modificar el diseño, ¿Es posible facilitar la tarea del programador de lenguaje ensamblador permitiéndole escribir `LCTE cte8` que tiene como efecto $R_0 \leftarrow cte8$ (donde cte8 es una constante de 8 bits)? Explicar cómo o justificar por qué no.