Práctica 3 - Arquitectura del CPU

Sistemas Digitales

Segundo Cuatrimestre 2024

Para realizar la siguiente guía de ejercicios se requiere hacer una la lectura del manual de RISC-V. El mismo se puede encontrar en el campus de la materia en la sección *Bibliografía*.

1. Ensamblado, compilación y seguimiento

Ejercicio 1 ¿Qué es una arquitectura? ¿Qué componentes la conforman? ¿Contiene información del funcionamiento interno de las operaciones?

Ejercicio 2

a) ¿A cuántos bytes se direcciona la memoria en la arquitectura RISC-V? ¿Cuántos bytes hay en una palabra?

b) Sabiendo que la memoria se encuentra en el estado que se ve a continuación, indicar el resultado de las siguiente operaciones sabiendo que `t0` = 0xAD.

Direccion	...	0xAA	0xAB	0xAC	0xAD	0xAE	0xAF	0xB0	0xB0	0xB0	...
Valor	...	0x34	0x11	0xF4	0x09	0x12	0x73	0x20	0x24	0xFF	...

a) **lw** `t1`, 0(`t0`) b) **lw** `t1`, 2(`t0`) c) **lw** `t1`, -3(`t0`) d) **lh** `t1`, -1(`t0`)

e) **lhu** `t1`, -1(`t0`) f) **lb** `t1`, 5(`t0`) g) **lbu** `t1`, 5(`t0`)

Ejercicio 3 Dado el siguiente arreglo de enteros de 16 bits en lenguaje Java:

```
1    int[] arreglo16b = {-1, 170, 255, -255, 0, 32, 10000, 0};
```

Sabiendo que este arreglo se guarda en memoria empezando en la direccion 0xCC.

a) Dibujar el estado de la memoria

b) Si `t0` = 0xCC, escribir un programa que dado un index i, devuelve **arreglo16b**[i]

Ejercicio 4 Dado los siguientes dos programas y suponiendo que ambos empiezan en la dirección 0x00.

a) Escribir en qué dirección se encuentra cada etiqueta

b) ¿Cómo se maneja el branching en RISC-V? ¿Y los saltos incondicionales? ¿Es afectado por la dirección de memoria donde comienza el programa? Para cada instrucción de salto, escribir el offset que se aplicará al PC

```
1  Inicio:  addi a0, zero, 10
2           addi a1, zero, 50
3  Ciclo:   ble a1, zero, Fin
4           sub a1, a1, a0
5           j Ciclo
6  Fin:     beq a1, zero, Inicio
```

```
1  Inicio:  li a1, 0xffffffff
2           li a2, 0x1
3  Vuelta:  beq a1, a2, Inicio
4           sub a2, a2, a1
5           nop
6           j Vuelta
```

Ejercicio 5 Dado el siguiente programa en lenguaje C.

```
1    int x = 2;
2    int y = 32;
3    x = x + y;
```

a) Traducir a lenguaje esamblador de RISC-V. Usar los registros `t0` y `t1` inicilizados con números de 8 bits para representar a las variables `x` e `y` respectivamente.

b) Escribir un programa que guarde en `t2` un número de 32 bits divido en sus 12 bits más significativos en `t0` y el resto de 20 bits en `t1`.

c) ¿Cómo maneja RISC-V la extensión de signo en los inmediatos de 12 bits? ¿Qué resultado generaría la instrucción **andi** `a0`, -2048 cuando `a0` vale 0xFFFFFFFF? Re-escribir el código del inciso **a)** para números de 32 bits sin utilizar la instrucción **li**.

Ejercicio 6

a) ¿Cuántos bytes ocupa cada instrucción de RISC-V? ¿Cuál es la diferencia entre una instrucción y una pseudoinstrucción?

b) ¿Qué clases de instrucciones tiene la arquitectura RISC-V? ¿Qué tipo de instrucciones contiene cada clase? ¿Qué diferencia hay entre instrucciones de Registros(**R**) y de Inmediatos(**I**)?

c) Ensamblar el siguiente código escrito en lenguaje RISC-V

```
1   addi a6, x0, 10
2   add a0, a1, a6
3   bltz x1, 0x0ABC
```

d) Desensamblar el siguiente programa escrito en lenguaje de máquina RISC-V.

```
0111 1111 1111 0000 0000 0101 0001 0011
0101 0101 0101 0000 0000 0101 1001 0011
0000 0000 1010 0101 1100 0110 0011 0011
1111 1110 0000 0110 0000 1010 1110 0011
0000 0000 0000 0000 0000 0000 0001 0011
```

Ejercicio 7 Responder las siguientes preguntas y luego realizar el inciso **c)**

a) ¿Qué registros contiene la arquitectura RISC-V y cuántos bytes de almacenamiento tiene cada uno? ¿Qué utilidad cumple tener diferentes clases de registros?

b) ¿Qué pasos conforman al ciclo de instrucción? ¿De qué se ocupa cada etapa y con qué componentes interactúa?

c) A partir de cada uno de los siguientes vuelcos parciales de memoria y estados del procesador, realizar un seguimiento simulando ciclos de instrucción. Para el primer caso, realizarlo hasta encontrar la instrucción ecall. Para el segundo caso, hasta hallar una instrucción inválida (es decir, una instrucción cuyo código de operación no figure entre los del conjunto de instrucciones). Indicar qué celdas de memoria se modifican y el estado final del procesador.

I)

pc	zero	00000000	gp	CCFBBFAA	tp	000000A3	t0	00000006	t1	00000087
00000000	t2	0000AA00	t3	000000B5	t4	000000BC	t5	00000073	t6	00000037
ra	a0	000034AA	a1	000000A0	a2	00000088	a3	00001CE6	a4	0000C2FC
00000000	a5	0000C2FC	a6	000049CB	a7	00008D83	s0	00000B01	s1	00000CF4
sp	s2	000004A6	s3	0000066A	s4	00000330	s5	00000071	s6	00000030
FFFFAA00	s7	00000077	s8	0000003B	s9	000000ED	s10	00000081	s11	006B23CD

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	00	00	03	13	00	10	03	93	00	72	CE	63	00	13	FE	13
0010	00	0E	04	63	00	80	00	6F	00	73	03	33	00	13	83	93
0020	FE	9F	F0	6F	00	00	00	73	00	00	00	00	00	00	00	00

II) *Ayuda*: la sexta instrucción a ejecutar es inválida.

pc	zero	00000000	gp	CCFBBFAA	tp	000000A3	t0	4B2BC396	t1	7B3E3D4A
0000BBB0	t2	50EEB50E	t3	0000BBC4	t4	0000BBC8	t5	BE5FC0FD	t6	1FC9F40C
ra	a0	000000AC	a1	000000FF	a2	6F2E1796	a3	E7C3495F	a4	4683A1D1
00000000	a5	04E68D53	a6	63068886	a7	DC136ADE	s0	C4BD2152	s1	BBF60FF6
sp	s2	AB0BD12A	s3	1C2357CE	s4	347D7720	s5	B3869CEB	s6	A947722F
FFFFAA00	s7	7FC4685E	s8	3DD38820	s9	2B39A78B	s10	385ADEF2	s11	22C6598A

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
BBB0	00	0E	22	83	00	0E	A3	03	00	55	05	33	00	65	85	B3
BBC0	00	A3	03	B3	AF	FB	CA	77	0A	70	25	23	00	00	00	13

2. Programación en RISC-V

Ejercicio 8 ¿Qué es .text y .data? ¿Qué tipo de información se guarda en cada una de ellas?

Ejercicio 9 Se cuenta con cuatro datos sin signo de un byte cada uno almacenados en el registro `t0` y queremos sumar el valor de los cuatro datos. Escribir un programa en lenguaje ensamblador RISC-V que realice esta operación y almacene el resultado en el registro `t0`.

Ejemplo:

`t0`	0x90	0x1A	0x00	0x02

Con este dato el registro debería valer 0x000000AC.

Ejercicio 10 En la arquitectura RISC-V tenemos una instrucción denominada `sll` que, dado un registro destino y dos registros fuente, mueve el primer registro fuente tantos bits a izquierda como indique el segundo registro fuente y guarda el resultado en el registro de destino. Por ejemplo, tenemos un dato en `t0` y un valor en `t1`:

`t0`	0x08	0x2B	0x00	0x23
`t1`	0x00	0x00	0x00	0x04

Luego de hacer `slli t0, t0, t1`, el registro `t0` quedaría de la siguiente forma:

t0	0x82	0xB0	0x02	0x30

Suponiendo que el valor a shiftear se encuentra en el registro `t0`, que el resultado se guarda en `t0` y que la cantidad de posiciones se encuentra en el registro `t1` (entendido como un número entero sin signo), se pide:

a) Escribir el *pseudocódigo* del programa `sll` asumiendo que no tenemos dicha instrucción disponible.

b) Escribir el programa de `sll` en lenguaje *assembler* de RISC-V.

c) El programa creado, ¿usa otros registros además de `t0` y `t1`? Si lo hace, modificar el programa de modo que solo se alteren los valores de `t0` y `t1`.

Ejercicio 11 Dado vector de enteros *Arreglo* y su *Longitud*, escribir un programa que encuentre el valor máximo del *Arreglo*. Se puede asumir que el inicio de del arreglo está en `t0` y la longitud en `t1`.

```
 Ejemplo:
- Entrada: Arreglo = [3, 1, 4, 1, 5, 9, 2, 6], Longitud = 8
- Salida: 9
```

Ejercicio 12 Sean dos vectores s y q tales que las direcciones de inicio se encuentran en `t0` y `t1` respectivamente. Sabiendo que la longitud de ambos se encuentra en `t2`, escribir un programa que copie la información de q a s.

Ejercicio 13 Sean dos vectores s y q tales que las direcciones de inicio se encuentran en `t0` y `t1` respectivamente. Sabiendo que la longitud de ambos se encuentra en `t2`, escribir un programa que copie los elementos pares de q a s. Si la posición i de q no tiene un elemento par, se debe poner un 0 en s.

Ejercicio 14 La arquitectura RISC-V posee operaciones aritméticas sobre números enteros codificados en notación *complemento a 2* de *32 bits*. El programa sumar64 realiza la suma en notación *complemento a 2* de dos números enteros de *64 bits* en esta arquitectura. En los registros `t0` y `t1` se indican las direcciones de cada número y en `t2` se indica la posición de memoria en donde debe guardarse el resultado.

Se pide:

a) Escribir el *pseudocódigo* del programa `sumar64`.

b) Escribir sumar64 en código *assembler* de RISC-V.

Ejercicio 15 `sumaVector64` es un programa que suma los valores de un vector de n posiciones de enteros de *64 bits* codificados en notación *complemento a 2*. En `t0` se recibe la cantidad de elementos que tiene el vector y en `t1` la posición de memoria en donde está almacenado dicho vector. En `t2` se recibe la posición de memoria en donde debe guardarse el resultado.

Suponiendo que se cuenta con el programa sumar64 descripto en el ejercicio anterior, se pide:

a) Escribir el *pseudocódigo* del programa sumaVector64.

b) Escribir sumaVector64 en código *assembler* de RISC-V.

3. Para pensar en otras arquitecturas (opcional)

Las diferentes arquitecturas que existen suelen variar en sus características. Por ejemplo, en el tamaño de una palabra, en la cantidad de instrucciones, en sus tipos o en el tamaño (fijo o variable). A continuación, dejamos una serie de ejercicios para explorar algunas de estas posibilidades.

Ejercicio 16 Dada una arquitectura con palabras de *32 bits*; decidir 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 *"media palabra"*.

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

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

Ejercicio 17 Dada una arquitectura con palabras e instrucciones de b *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 intrucciones incluyen sólo un operando con modo de direccionamiento directo a memoria?

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

Ejercicio 18 ¿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 19 Dada una máquina con instrucciones de *16 bits* y direcciones de *4 bits*, diseñar 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 20 Diseñar un formato de instrucción con código de operación extensible (la cantidad de *bits* del código de operación 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 21 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.