PROBLEMAS 2

EL REPERTORIO DE INSTRUCCIONES

■ 1. Disponemos de un computador de instrucción única, es decir, su repertorio se reduce a la instrucción: *resta y salto si negativo*. Esta instrucción tiene tres operandos que son, cada uno de ellos, direcciones de una palabra en memoria:

rsn ***a***, ***b***, ***c***

Su funcionalidad es la siguiente: mem[***a***] = mem[***a***] - mem[***b***];
si (mem[***a***]<0) entonces ir a ***c***

La instrucción restará el número de la localización de memoria ***b*** del número en la localización de memoria en ***a*** y situará el resultado en a, sobrescribiendo el valor previo. Si el resultado es mayor o igual a 0, el computador tomará la siguiente instrucción de la localización de memoria inmediatamente posterior a la instrucción actual. Si el resultado es menor que 0, la siguiente instrucción se toma de la localización de memoria ***c***. Este computador no dispone de registros.

Escribe un programa para este computador que copie el valor contenido en la localización de memoria ***a*** en la localización de memoria ***b***.

■ 2. El formato del repertorio de instrucciones de un procesador se ha diseñado con el objetivo de minimizar el espacio de representación. De esta manera, se han especificado dos formatos para los códigos de operación: uno "regular" y otro "extendido". El formato "regular" proporciona las operaciones sobre un campo de 6 bits mientras que el "extendido" utiliza 8 campos de extensión, de entre los posibles códigos de 6 bits del formato "regular", para llamar a un campo código de operación extendido de 3 bits. Sabemos que el formato "regular" se usa aproximadamente un 90% de las ocasiones en los programas de *test* de los que disponemos.

Sabiendo todo esto responder a las siguientes preguntas:

a) número de instrucciones del formato "regular"
b) número de instrucciones del formato "extendido"
c) número total de instrucciones distintas que se pueden codificar con este sistema
d) tamaño del campo código de operación en el caso de utilizar únicamente el formato regular para el número de instrucciones que maneja este procesador
e) tamaño medio del formato para las aplicaciones de *test*
f) tamaño medio si la distribución de instrucciones fuera equitativa
g) porcentaje de uso del formato "extendido" que hace ineficaz el diseño.

■ 3. Sea una máquina de acumulador (Acc) de la que se quieren diseñar los formatos de representación de sus instrucciones teniendo en cuenta las siguientes características del computador:

- el tamaño de palabra es de 16 bits;
- el modelo de ejecución es registro-registro;
- la memoria principal es de 2 Kpalabras;
- tiene 32 registros, entre los cuales se encuentra el acumulador (Acc), el PC, y el SP;
- los modos de direccionamiento que puede tener el procesador son: inmediato, directo a registro y a memoria, relativo a registro base y relativo a PC;
- las operaciones que debe poder ejecutar son:
  - instrucciones de una dirección:
    - STA: almacena el contenido del Acc en un registro o en una dirección de memoria.
    - LDA: carga en el Acc un inmediato, el contenido de una posición de memoria o el de un registro.
    - Bc: salto condicional (las posibles condiciones son Z, NZ, C, NC y el único modo de direccionamiento para esta instrucción es relativo a PC).
    - BR, CALL, PUSH, POP, ADD, SUB, MUL, AND, OR, XOR, CMP
  - instrucciones de cero direcciones:
    - NEG, INC, DEC, NOT, RET, SHL, SHR, ROL, ROR, HALT, WAIT

Sabiendo todo esto, diseñar el/los formatos de representación de instrucciones, indicando los modos de direccionamiento que puede admitir cada instrucción (o grupo de instrucciones), y minimizando el espacio de representación.

PROBLEMAS 2

■ 4. Contamos con máquinas de diferentes arquitecturas que podemos agrupar, estudiando sus juegos de instrucciones, de la siguiente forma:

ARQUITECTURAS					
Formato	0 direcciones	1 dirección	2 direcciones	3 direcciones	
Modo de ejecución	pila	acumulador	registro-registro	memoria-memoria	registro-registro (carga/almacenamiento)

Juego de instrucciones	PUSH M
	POP M
	ADD
	SUB
	MUL
	DIV
		LOAD M
		STORE M
		ADD M
		SUB M
		MUL M
		DIV M
			LOAD X, M
			STORE M, X
			MOVE X, Y
			ADD X, Y
			SUB X, Y
			MUL X, Y
			DIV X, Y
				ADD M1, M2, M3
				SUB M1, M2, M3
				MUL M1, M2, M3
				DIV M1, M2, M3
					LOAD X, M
					STORE M, X
					MOVE X, Y
					ADD X, Y, Z
					SUB X, Y, Z
					MUL X, Y, Z
					DIV X, Y, Z

Los operandos designados con M son posiciones de memoria mientras que los designados con X, Y o Z se refieren a registros del procesador.

La máquina de 0 direcciones tiene un modo de ejecución a pila de forma que todas las operaciones se realizan en la cima de la pila. Las operaciones de proceso toman sus operandos de la cima de la pila y los eliminan sustituyéndolos por el resultado. Las operaciones de transferencia (PUSH y POP) son las únicas que realizan accesos a memoria.

La máquina de 1 dirección tiene un modo de ejecución basado en acumulador. Las operaciones de proceso se realizan con un operando de memoria y otro implícito que es siempre el acumulador. El juego incluye dos instrucciones de transferencia para cargar el registro acumulador (LOAD M) o para almacenar su contenido en memoria (STORE M).

La máquina de 2 direcciones realiza las operaciones de proceso entre dos operandos residentes en alguno de los 16 registros con que cuenta. Las instrucciones de transferencia son las encargadas de mover información entre los registros y memoria (LOAD, STORE) o entre los propios registros.

Las máquinas de 3 direcciones especifican tres operandos por instrucción. Se proponen en la tabla dos casos extremos: uno de ellos trabaja en modo de ejecución memoria-memoria mientras que el otro es registro-registro (también llamado de carga/almacenamiento).

Sabiendo todo esto, escribir los programas correspondientes a cada juego de instrucciones propuesto de forma que realicen el siguiente cálculo:

$$A = ((B + C) \cdot D) / (E – F \cdot G – H \cdot I)$$

Partir de la situación en la que todas las variables están en memoria.

■ 5. Medir la eficiencia de memoria de las diferentes arquitecturas propuestas en el problema anterior. Se harán las siguientes suposiciones sobre los repertorios de instrucciones:

- los códigos de operación son de 1 byte
- el direccionamiento de los operandos de memoria ocupa 2 bytes
- los operandos son de 2 bytes
- los registros se especifican mediante campos de 4 bits (16 registros)

Calcular para cada programa dado en la solución del problema anterior el número de bytes que ocupa y el número de bytes que se transfieren a través del bus de datos. Determinar cual es la arquitectura más eficiente si solo se tiene en cuenta el tamaño del código ejecutable. Determinar la eficiencia si se evalúa desde el punto de vista del ancho de banda total de memoria que se necesita, es decir, la suma de bytes de código más datos que se han de mover.

PROBLEMAS 2

■ 6. Un procesador ejecuta saltos condicionales (BR *cc*) utilizando el modo de direccionamiento relativo al PC. Los saltos incondicionales (JMP), sin embargo, solamente admiten el modo de direccionamiento absoluto a memoria. El procesador genera direcciones de 32 bits y los accesos a memoria están alineados a palabras de 32 bits, es decir, los dos bits de menor peso son siempre '00'. El formato de instrucción del salto condicional relativo a PC es el siguiente

donde la condición de salto se evalúa sobre el registro especificado en el campo registro (RA). Explíquese por qué el siguiente código hace que el compilador genere error:

```
aquí:   BRZ r1, allí
        ::::::::::::::::
        ::::::::::::::::
allí:   ADD r1, r1, r1
```

siendo las posiciones de memoria etiquetadas las siguientes:

aquí ➔ 01 00 00 00h
allí ➔ 08 00 00 00h

y BRZ es la instrucción de salto si es cero el registro, en este caso r1.

¿Cómo se puede solucionar este problema?