Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (II/2016)

Solución Interrogación 2

Pregunta 1

a) Luego de recibir la señal `INTA`, ¿qué tarea(s) debe realizar un controlador de interrupciones? **(1 pto.)**
Solución: Una vez recibida la señal, el controlador de interrupciones debe revisar en sus registros internos cuál fue el dispositivo que generó la IRQ. Una vez obtenido este ID, es enviado a la CPU a través del bus de datos.

b) Dos computadores poseen la misma ISA, pero uno tiene microarquitectura Von Neumann, mientras que el segundo es Harvard. ¿Es posible ejecutar el mismo programa en ambos computadores, sin realizar modificaciones a este? **(1 pto.)**
Solución: Sí, si ambos computadores usan la misma ISA, la ejecución del programa será independiente de la microarquitectura.

c) Implemente, utilizando sólo la instrucción `SUBLEQ a,b,c`, la instrucción `SUB a,b`. **(1 pto.)**
Solución: `SUB a,b`

- dir0: `SUBLEQ a,b,dir1`
- dir1: ...

d) ¿Por qué motivo podría dejar de cumplirse la ley de Moore en el futuro cercano? **(1 pto.)**
Solución: Si el proceso de miniaturización sigue desarrollandose al ritmo actual, el tamaño de los transistores llegará dentro de poco al límite físico para el tamaño de una compuerta lógica.

e) ¿Que ventaja(s) tiene el mapear un dispositivo de I/O a memoria? **(1 pto.)**
Solución: Un dispositivo de I/O mapeado a memoria puede ser utilizado en conjunto con el DMA para permitir transferencias desde/hacia memoria que no requieran la participación de la CPU.

f) Al finalizar la siguiente sección de código, ¿es el contenido de los registros AX y BX el mismo? **(1 pto.)**

```
MOV [0],AX
MOV BH, [0]
MOV BL, [1]
```

Solución: No, ya que la arquitectura x86 utiliza endianness little-endian, por lo que al guardar el valor en BX, el orden de los bytes estará invertido, si se compara con AX.

Pregunta 2

a) ¿Es posible soportar subrutinas en el computador básico, si se eliminan de la ISA las instrucciones `CALL` y de salto? Justifique su respuesta. **(1 pto.)**
Solución: Si es posible. Basta con utilizar la instrucción RET para emular la instrucción CALL de la siguiente forma:

I. Se pone un label (label_ret) después de la llamada a la subrutina para tener el valor (PC + 1).
II. Se guarda el valor del label en algún registro (por ejemplo MOV A, label_ret) y se hace push al stack (PUSH A).
III. Se pone un label (label_call) antes del código de la subrutina para así poder entrar a esta.
IV. Se guarda el valor del label en algún registro (por ejemplo MOV A, label_call) y se hace push al stack (PUSH A).
V. Se ejecuta la instrucción RET para saltar al ultimo valor guardado en el stack (la linea donde comienza la subrutina).

b) Modifique (sólo) la ISA del computador básico para soportar la instrucción `CALL reg`, que permite llamar a la subrutina ubicada en la dirección de memoria almacenada en el registro `reg`. **(2 ptos.)**
Solución: Se puede reservar una dirección de memoria para poder guardar el valor del registro en esta dirección (llamémosla dir, esta dirección es fija y se utilizará siempre que se ejecute la instrucción CALL reg). Luego, se debe agregar la instrucción CALL (a), que es equivalente al CALL normal, solo que el valor al cual se saltará no viene dado por un literal desde la memoria de instrucciones, sino que viene de la memoria principal, en la posición (a), por lo que la instrucción se debe realizar en 2 ciclos: Uno para guardar en la memoria principal el valor del PC + 1 antes de entrar a la subrutina, y otro para obtener el valor al cual se quiere saltar desde la dirección dir en la memoria principal (Esto es soportado por el computador básico ya que el MUX PC permite cargar al PC por un literal proveniente de la memoria de instrucciones o por un valor de la memoria principal).

Las señales de instrucción para soportar CALL (a) son:

Lpc	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0,1	Sdin0	Spc0	W	IncSp	DecSp
0	0	0	-	-	-	SP	PC	-	1	0	1
1	0	0	-	-	-	LIT	-	MEM	0	0	0

Finalmente, para implementar CALL reg basta con definirla como una instrucción de 3 ciclos como sigue:

I. MOV (dir), reg
II. CALL (dir)

c) Describa un mecanismo para, **en tiempo de ejecución**, escribir el código de una subrutina y luego llamarla, utilizando el assembly x86 de 16 bits. Asuma que tiene disponible la especificación completa de la ISA. **(3 ptos.)**
Solución: La arquitectura x86 es autoprogramable, ya que está basado en la arquitectura Von Neumannm, en donde la memoria principal se utiliza tanto para guardar instrucciones como para almacenar datos. Así, si se conoce toda la especificación de la ISA, es posible escribir programas que escriban programas, en particular subrutinas, de la siguiente forma:

I. Se guarda úna dirección de memoría específica en donde comenzará a escribirse la subrutina.
II. Desde esa dirección, se comienzan a escribir números de manera contigua en memoria que corresponden a los opcodes de la subrutina.

III. Al terminar de escribir la subrutina en memoria, se hace CALL a la dirección en donde comienza la subrutina.

Pregunta 3

Para los siguientes ejercicios, considere la siguiente tabla, que presenta el vector de interrupciones completo de un computador con ISA x86 de 16 bits. El vector de interrupciones se encuentra almacenado a partir de la dirección de memoria `0x0000`:

IRQ	Dispositivo	Pos. en vector
IRQ0	Timer del sistema	00
IRQ1	Disco Duro	01
IRQ2	Interfaz USB	02
IRQ3	Interrupción software	03

a) ¿Cuantos dispositivos que generen solicitudes de interrupción pueden conectarse? **(1 pto.)**
Solución: Debido a que se puede conectar un número arbitrario de dispositivos en la controladora USB, es infinito la respuesta.

b) Dos dispositivos, teclado y mouse, están conectados a la interfaz USB. Describa un mecanismo para ejecutar la ISR correspondiente al mouse, cuando este genera una interrupción. **(1 pto.)**
Solución: Un mecanismo que opera es que al generar una interrupción, se llama a la ISR de la controladora USB, ésta lee algún registro de estado de la controladora para determinar la rutina que debe seguir atendiendo y la invoca, ya sea como Trap, subrutina, etc.

c) ¿Que ocurriría en este computador si se ejecuta la instrucción `MOV [0],AX`? **(1 pto.)**
Solución: Se cambiaría el puntero a la ISR del timer del sistema, no confundir con cambiar el timer. Se puede inferir que esto sería fatal para el funcionamiento del computador.

d) Proponga un esquema para permitir el acceso (lectura y escritura) controlado y centralizado al vector de interrupciones por parte de los programas, *i.e.*, el acceso sólo puede realizarse a través de una interfaz entregada por el sistema operativo (o la BIOS). **Hint**: El esquema puede incluir cambios a la arquitectura del computador. **(3 ptos.)**
Solución: Una posible solución es transportar el vector de interrupciones a un registro especializado, que sólo puede ser escrito en modo supervisor y es accesado por nuevas instrucciones para escritura, por ejemplo: SINT <ID Interupción>, <Label ISR = lit>. La ejecución seguiría usando INT en la ISA, pero su camino de ejecución debe ser alterado.