ESTRUCTURA DE COMPUTADORES. Grado en Matemáticas e Informática
PRIMER PARCIAL (26 de marzo de 2019)

---

1 *(2 puntos) Dada la instrucción de una palabra CALL [.R1]*

a) *Describa las distintas fases de ejecución de la instrucción e indique los registros que se modifican en cada una de ellas.*

b) *Suponiendo que la instrucción está en la posición de memoria 1000 y que el puntero de pila contiene el valor 3000, describa el contenido de la pila una vez ejecutada la instrucción, así como el valor del puntero de pila. Suponga para ello que las palabras y direcciones son de 32 bits, el direccionamiento es a nivel de byte, y el registro puntero de pila apunta a la primera posición libre de esta, que crece hacia direcciones de memoria decrecientes*

SOLUCIÓN

a) Fases de ejecución de la instrucción

En primer lugar está la fase de Fetch, que realiza un acceso a memoria a la dirección donde apunta el contador de programa e incrementa éste. Lo que viene de memoria (la instrucción) se mete en el registro de instrucción.

Se realiza la decodificación.

La ejecución consta de:

- Un segundo acceso a la memoria, esta vez de escritura a la dirección donde apunta el puntero de pila y un decremento de éste. Lo que se escribe es el contenido del contador de programa (dirección de retorno).
- Carga del contador de programa con el contenido del registro R1, lo que significa que la siguiente instrucción que se ejecutará será la que esté almacenada en dicha dirección.

b) Contenido de la pila

En la dirección 3000 queda almacenado el valor 1004 (PC+4) y el puntero de pila pasa a contener 2996 (SP-4).

2 *(3 puntos) A continuación se muestra el código, en ensamblador IEEE, de una subrutina a la que se llama desde un programa principal. Todas las instrucciones ocupan una palabra. El computador donde se ejecuta tiene palabras y direcciones de 32 bits y direccionamiento a nivel de byte. Dispone además de un registro puntero de pila (SP) que apunta a la primera posición libre de esta, que crece hacia direcciones de memoria decrecientes.*

```
Subrutina
      AND .R10, #0
      LD  .R3, #12[.SP]
      LD  .R4, #8[.SP]
      CMP [.R3++], [.R4++]
      BZ $16
      ADD .R10, #1
      CMP .R10, .R2
      BZ $4
      BR $-24
      RET
```

a) *Rellene la tabla que se proporciona en la hoja adjunta, con los valores sucesivos que van tomando los registros y las posiciones de memoria afectadas por la ejecución de cada una de las instrucciones de la subrutina, incluyendo el biestable Z (no es necesario que incluya el PC, salvo en la instrucción RET). Considere para ello lo siguiente:*

- ***El contenido de la pila*** *al inicio de la ejecución de la subrutina es el que se muestra en la hoja adjunta.*
- ***El registro R2*** *tiene el valor 5, las direcciones de memoria H'400 hasta H'410 contienen los valores 1, 2, 3, 4 y 5, y las direcciones H'500 hasta H'510 contienen los valores 0,2,4,6,8.*

Indique en la última columna el número y tipo de accesos a memoria que se realizan en la ejecución de cada una de las instrucciones.

b) *Razone si se podrían sustituir las instrucciones `LD` de la subrutina por instrucciones `POP` de los registros correspondientes.*

c) *Si el procesador tuviese únicamente modelo de ejecución registro-registro, indique razonadamente qué instrucciones de la subrutina no serían válidas. Sustituya estas instrucciones por la secuencia de instrucciones equivalente. Para ello puede utilizar los registros auxiliares `RT1`, `RT2` y `RT3`.*

d) *Suponga ahora que en la pila en lugar del valor 500 se tuviera el valor 502 ¿Se produciría algún error si el procesador no admitiese accesos a información no alineada? Si la respuesta es afirmativa, ¿en cuál de las instrucciones se produciría el error? Justifique su respuesta.*

SOLUCIÓN

a) En la tabla adjunta se muestra el orden de ejecución de las instrucciones, los registros afectados por su ejecución y los accesos a memoria que se realizan. Aparte de los accesos mostrados en la tabla, en todas las instrucciones hay un acceso de lectura para leer la instrucción (*fetch*), ya que todas ocupan solo una palabra.

b) No se pueden sustituir las instrucciones LD por POP, por una parte porque POP modifica el valor del puntero de pila mientras que LD no lo hace y por otra parte porque no se cargarían los mismos valores en R3 y en R4, por lo que la subrutina no haría lo mismo.

POP .R3 leería de la pila el valor 112 (que corresponde a la dirección de retorno) y lo almacenaría en R3, incrementando posteriormente el SP, y POP .R4 leería de la pila el valor 500 incrementando de nuevo el SP. Cuando se ejecutase la instrucción RET se sacaría de la pila el valor 400, que supuestamente debería ser la dirección de retorno, por lo que se retornaría a una dirección incorrecta.

c) La única instrucción no válida en el modelo de ejecución registro-registro es CMP [.R3++], [.R4++] ya los dos operandos están en memoria y no en registros. La secuencia de instrucciones equivalente, que sigue el modelo registro-registro, podría ser la siguiente:

```
LD  .RT1, [.R3++]
LD  .RT2, [.R4++]
CMP .RT1, .RT2
```

d) La instrucción LD .R4, #8[.SP] cargaría en R4 el valor 502 sin problema, pero al ejecutarse la instrucción CMP [.R3++], [.R4++] se intentaría acceder a una palabra cuya dirección, 502, no es múltiplo de 4, que es el número de bytes que ocupa cada palabra, por lo que se produciría una excepción en la ejecución de esta instrucción.

	R2	R10	R3	R4	Z	PC	SP	Accesos
	5							
AND R10, #0		0						
LD R3, #12(SP)			400					1 acceso de lectura
LD R4, #8(SP)				500				1 acceso de lectura
CMP (R3++), (R4++)			404	504	0			2 accesos de lectura
BZ $16								
ADD R10, #1		1						
CMP R10, R2					0			
BZ $4								
BR $-24								
CMP (R3++), (R4++)			408	508	1			2 accesos de lectura
BZ $16								
RET						112	Sp+4	1 acceso de lectura

3 *(5 puntos) Un computador cuenta con un formato de representación de números en coma flotante de 16 bits. El bit superior representa el signo del número, los 9 siguientes el exponente y los 6 últimos la mantisa. Este formato sigue las convenciones del formato estándar IEEE754 en todo excepto en los tamaños, es decir, usa el mismo tipo de representaciones especiales, representación del exponente, bit implícito, situación de la coma, representaciones normalizadas y no normalizadas, así como bits de guarda y redondeo.*

a) *Determine el rango y resolución del formato*

b) *Represente en el formato los números decimales* $A = 50,25$ *y* $B = -0,1$

c) *Dadas las cadenas* $C = H'BFDB$ *y* $D = H'800D$ *que son números representados en el formato, determine su valor decimal*

d) *Realice la operación* $A + C$*, dejando el resultado en el formato de partida y determinando su valor decimal. Utilice redondeo al más proximo.*

e) *Determine el número de bits que debería tener la mantisa para que el número decimal* $0,1$ *se pueda representar con una resolución mejor que* 10^{-3}*.*

SOLUCIÓN

a) Rango y resolución del formato.

Números normalizados.

Exponente: El estándar IEEE754 representa los exponentes en exceso a $2^{n-1}-1$. En este caso sería exceso a 255. Como se reserva el exponente mínimo (-255) para la representación del cero y los números no normalizados, y el exponente máximo ($+256$) para la representación del infinito y las indeterminaciones, el rango de exponente para la representación de números queda: $[-254, 255]$.

La mantisas normalizadas en este formato se representan en signo-magnitud, con bit implícito y la coma situada a la derecha del bit implícito:

$$\text{Mantisa: } \pm \begin{cases} \texttt{1,000000} & \rightarrow & 1 \\ \texttt{1,111111} & \rightarrow & 2 - 2^{-6} \end{cases}$$

El rango para números normalizados es: $\pm \left[1 \cdot 2^{-254}, (2 - 2^{-6}) \cdot 2^{255}\right]$

Números no normalizados.

Exponente: Siguiendo el estándar IEEE754, aunque los números no normalizados se representan con el exponente todo a ceros (valor -255), todos ellos tienen como exponente el más pequeño de los normalizados, en este caso -254. De este modo hay continuidad en la representación.

$$\text{Mantisas: } \pm \begin{cases} \texttt{0,000000} & \rightarrow & 0 \\ \texttt{0,000000} & \rightarrow & 2^{-6} \\ \texttt{0,111111} & \rightarrow & 1 - 2^{-6} \end{cases}$$

El rango para números no normalizados es: $\pm \left[2^{-6} \cdot 2^{-254}, (1 - 2^{-6}) \cdot 2^{-254}\right] \cup 0$

La resolución depende del exponente y es: $2^{-6} \cdot 2^{E}$

b) Representación del número.

$A = \texttt{+50,25} = \texttt{+110010,01} = \texttt{+1,100100} \cdot 2^5 = \texttt{0 100000100 100100} = \texttt{H'4124}$

$B = \texttt{-0,1} = \texttt{-,000110011001} = \texttt{-1,100110} \cdot 2^{-4} = \texttt{1 011111011 100110} = \texttt{H'BEE6}$

c) Valor decimal.

$C = \texttt{H'BFDB} = \texttt{1 011111111 011011} = \texttt{-1,011011} \cdot 2^0 = \texttt{-1,011011} = \texttt{-1,421875}$

$D = \texttt{H'800D} = \texttt{1 000000000 001101}$

El exponente es todo ceros, luego estamos ante una representación no normalizada. Por tanto ponemos el bit implícito a cero y el exponente a -254

$D = \texttt{-0,001101} \cdot 2^{254} = \texttt{-13} \cdot 2^{260}$

d) Suma $A + C$.

Este formato utiliza dos bits de guarda y un bit retenedor para la suma. Los números a sumar son:

$A = \texttt{+1,100100} \cdot 2^5 \quad y \quad C = \texttt{-1,011011} \cdot 2^0$

Se restan los exponentes: $E_A - E_C = 5 - (0) = 5$. Hay que desplazar la mantisa de C cinco lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

$C = \texttt{-0,000010 111} \cdot 2^5$

M_A	$\texttt{1,100100 000}$	
M_C	$\texttt{− 0,000010 111}$	
	$\texttt{1,100001 001} \cdot 2^5$	*Normalizado*
Redondeo	$\texttt{0,000000 100}$	
	$\texttt{1,100001 101} \cdot 2^5$	

Como los tres bits que se usan como información de redondeo son 001 (no es el caso especial 100), el redondeo se realiza sumando un uno en la posición más alta de estos bits y truncando.

$A + C = \texttt{+1,100001} \cdot 2^5 = \texttt{0 100000100 1000001} = \texttt{H'4121}$

$A + C = \texttt{+110000,1} = \texttt{+48,5}$

e) Número de bits de la mantisa.

La resolución de este sistema de representación es $2^{-6} \cdot 2^E$ porque el número de bits de la mantisa es 6. En general será $2^{-n} \cdot 2^E$ siendo n el número de bits de la mantisa.

Hemos visto que el número 0,1 se representa con el exponente -4. Así:

$2^{-n} \cdot 2^{-4} < 10^{-3} \rightarrow 2^{-n-4} < 10^{-3} \rightarrow (n+4) \cdot log2 > 3$

$n + 4 > 3/log2 = 9,96 \rightarrow n = 6$

NOTAS: 23 de abril **DURACIÓN: 1 hora y 50 minutos**
REVISIÓN: 25 de abril **PUNTUACIÓN: especificada en cada ejercicio**