ESTRUCTURA DE COMPUTADORES (Grado en MI)

EXAMEN FINAL DE JULIO -parcial 2- (4 de julio de 2018)

PROBLEMAS

1 (4 puntos) *Programe la función* `intercambia_diagonales(matriz,n)` *en ensamblador del MC88110 que intercambia la diagonal principal con la diagonal secundaria de una* `matriz` *cuadrada de orden* `n` *almacenada por filas cuyos elementos son enteros de una palabra. Ambos parámetros se pasan en la pila:* `n` *por valor y* `matriz` *por dirección.*

A modo de ejemplo se muestra la ejecución de una llamada a `intercambia_diagonales` a la que se pasa como parámetro la matriz que aparece en la figura (de orden 4) y el resultado de la ejecución se indica a la derecha de este párrafo.

$$Matriz = \begin{pmatrix} \mathbf{1} & 2 & 3 & \mathbf{4} \\ 5 & \mathbf{6} & \mathbf{7} & 8 \\ 9 & \mathbf{10} & \mathbf{11} & 12 \\ \mathbf{13} & 14 & 15 & \mathbf{16} \end{pmatrix}$$

$$Resultado = \begin{pmatrix} \mathbf{4} & 2 & 3 & \mathbf{1} \\ 5 & \mathbf{7} & \mathbf{6} & 8 \\ 9 & \mathbf{11} & \mathbf{10} & 12 \\ \mathbf{16} & 14 & 15 & \mathbf{13} \end{pmatrix}$$

Para la realización de este ejercicio se llevará a cabo el tratamiento del Marco de Pila descrito en clase y se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura; que las subrutinas llamantes dejan disponibles todos los registros excepto r1, r30 (SP) y r31 (FP); que la pila crece hacia direcciones de memoria decrecientes y el puntero de pila apunta a la última posición ocupada (de la misma forma que se ha utilizado en el proyecto).

SOLUCIÓN

Para intercambiar las diagonales se utilizarán dos punteros que apuntan al elemento en curso de la diagonal principal (r20) y de la diagonal secundaria (r21). El desplazamiento que hay que aplicar a estos punteros en cada iteración es el número de direcciones que separan a dos elementos consecutivos de una diagonal. En el caso de la diagonal principal es (n+1)*4 y en el caso de la diagonal secundaria es (n-1)*4. El puntero de la diagonal principal se inicializa al comienzo de la matriz y el de la diagonal secundaria a la dirección del último elemento de la primera fila. A continuación se muestra el código comentado.

```
intercambia_diagonales:
        ld r2,r30,4
        addu r3,r2,1      ; Se calculan los dos desplazamientos:
        mulu r3,r3,4      ; En la diagonal ppal dos elementos están separados
        subu r4,r2,1      ; por (n+1)*4 bytes, mientras que en la secundaria por
        mulu r4,r4,4      ; (n-1)*4
        ld r20,r30,0      ; Se sitúan los punteros r20 y r21 al elemento de la diagonal
        addu r21,r21,r4   ; principal y secundaria respectivamente

bucle:  cmp r7,r2,r0      ; Se pregunta si se ha llegado al final
        bb1 eq,r7,fin
        ld r5,r20,r0      ; Se intercambian los elementos de las diagonales
        ld r6,r21,r0
        st r5,r21,r0
        st r6,r20,r0
        subu r2,r2,1      ; Decrementa el contador
        addu r21,r21,r4   ; Se incrementan los punteros con los desplazamientos
        addu r20,r20,r3   ; calculados al principio
        br bucle
fin:    jmp(r1)
```

2 (3 puntos) *Considere un controlador industrial basado en un procesador con palabras y direcciones de 32 bits y direccionamiento a nivel de byte. El sistema posee una memoria caché asociativa de 32 KBytes inicialmente vacía, con bloques de 16 bytes y política de escritura inmediata sin actualización (WTWNA). En este sistema se ejecuta el siguiente fragmento de un programa almacenado a partir de la dirección 0 en el que tanto las instrucciones como los datos ocupan una palabra:*

```
        add   r10, r0,  800
        add   r12, r0,  0
```

```
        add   r3,  r0,  20
buc:    ld    r20, r10, 0
        ld    r21, r10, 4
        add   r12, r12, r20
        sub   r12, r12, r21
        st    r12, r10, 0
        add   r10, r10, 8
        sub   r3,  r3,  1
        cmp   r4,  r3,  0
        bb1   gt,  r4,  buc
```

***a)* (60 %)** *Represente su traza de ejecución para las tres primeras iteraciones del bucle. Considerando la ejecución del fragmento de código completo, calcule el número total de accesos y el número de fallos que se producirán en los accesos a caché, distinguiendo entre los producidos en las instrucciones y en los datos. Especifique en qué direcciones de memoria se producen los fallos.*

***b)* (40 %)** *Calcule la tasa de aciertos y el tiempo de acceso efectivo a memoria sabiendo que el tiempo de acceso de la memoria principal es de 80 ns y el de la memoria caché de 4 ns.*

SOLUCIÓN

a) El fragmento de código consta de 3 instrucciones que se ejecutan una sola vez, seguidas de un bucle de 9 instrucciones que se ejecuta 20 veces. Los únicos accesos a datos se producen en la parte del bucle y se trata de tres accesos a direcciones consecutivas, lo que se traduce en un total de 60 accesos a datos. La traza de ejecución para las tres primeras iteraciones del bucle es la siguiente:

```
0   4   8  12 800(Rd) 16 804(Rd) 20 24 28 800(Wr) 32 36  40  44
           12 808(Rd) 16 812(Rd) 20 24 28 808(Wr) 32 36  40  44
           12 816(Rd) 16 820(Rd) 20 24 28 816(Wr) 32 36  40  44  ...
```

Como se ha indicado, se producen $3 + 9 \times 20 = 183$ accesos a instrucciones y 3×20 accesos a datos, de los que 40 son lecturas y 20 escrituras, en total 243 accesos. Durante la ejecución de instrucciones solo se producen 3 fallos, puesto que solamente se utilizan 12 instrucciones distintas y son consecutivas, estando almacenadas en tres bloques de caché. En el acceso a datos se utilizan 2 direcciones distintas en cada iteración, es decir, un total de 40 direcciones distintas, todas consecutivas, lo que provocará 10 fallos de caché, 30 aciertos de lectura y 20 aciertos de escritura.

Así pues, en total se producen 13 fallos de caché, que se dan en las direcciones siguientes:

```
Instrucciones: 0 .. 16 .. 32
Datos: 800 .. 816 .. 832 .. 848 .. 864 .. 880 .. 896 .. 912 .. 928 .. 944
```

b) La tasa de aciertos al ejecutar el programa del enunciado se deduce directamente de los cálculos realizados en el apartado anterior:

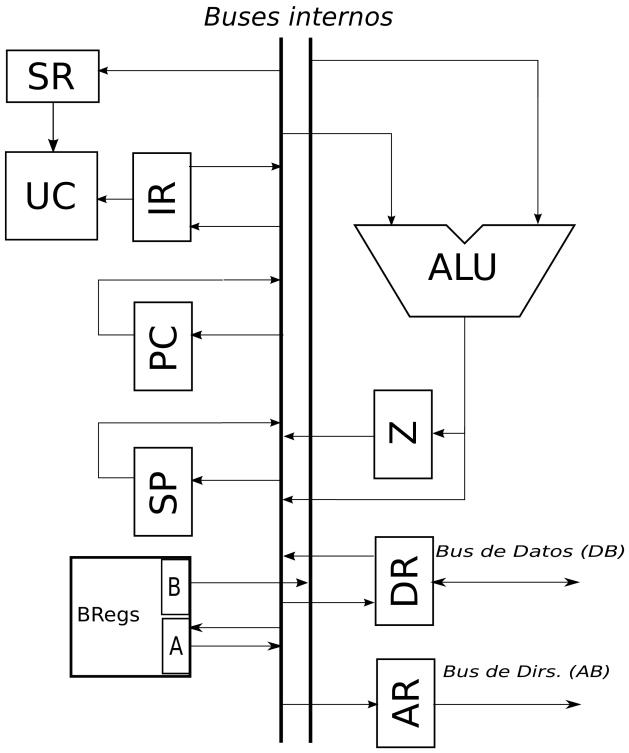
$Hr_{Mca} = N^o aciertos/N^o accesos = 1 - N^o fallos/N^o accesos = 1 - 13/243 = 0,946 \rightarrow 94,6\,\%$

Los accesos de lectura con acierto (183 instrucciones y 30 datos) se completan en 4 ns (t_{Mca}). Las escrituras (20 aciertos) necesitan 80 ns (t_{MP}), mientras que los fallos, que siempre se producen en lecturas, requieren que se lea un bloque de memoria principal. En total, el tiempo de acceso efectivo será:

$t_{acc} = ((180 + 30) \times 4 + 20 \times 80 + 13 \times (4 + 4 \times 80 + 4))\ ns/243\ accesos = 6.704\ ns/243 = 27,59$ ns

3 (3 puntos). *Sea la CPU cuyo esquema simplificado aparece en la figura. La ALU, todos los registros, rutas de datos y de direcciones son de 32 bits.*

PC: Reg. contador de programa
AR: Reg. de direcciones
IR: Reg. de instrucción
Z: registro transparente
BRegs: banco de registros de propósito general*, R0..R7*
SP: Reg. puntero de pila
DR: Reg. de datos
SR: Reg. de estado



a) *Suponiendo que:*

1. *el banco de registros dispone de dos puertas, A y B, que permiten a la UC seleccionar cualquier pareja de registros en cada ciclo.*
2. *cada instrucción ocupa una palabra.*
3. *el campo #desp de la instrucción se indica como IR.desp (campo del reg. de instrucción).*
4. *la memoria es direccionable a palabra y necesita para operar dos ciclos de reloj.*
5. *el tiempo de ciclo de reloj es 30 ns.*
6. *la pila se llena hacia direcciones decrecientes y SP apunta a la primera dirección libre.*

a.1) *Realice la descomposición en operaciones elementales, indicando claramente las acciones que se realizan en cada ciclo de reloj, para:*

1) el **fetch** *(común a todas las instrucciones.)*

2) las instrucciones que aparecen a continuación. Señale con **fetch** *la secuencia anterior, que se supondrá al principio de cada instrucción. Indique claramente con el texto* **ACTUALIZAR_SR** *los ciclos en que se deba actualizar el registro de estado, SR.*

```
1) ST .R3, #12[.R5]        2) ADD .R2, .R4, .R6        3) POP .R7
```

a.2) *En función del resultado del apartado anterior, indique en cada caso el número total de ciclos –incluido el fetch– que tardaría en ejecutarse cada instrucción y su equivalente en tiempo.*

a.3) *Indique si encuentra alguna posible modificación en esta estructura que permitiese reducir el número de ciclos necesarios para la ejecución de las instrucciones propuestas.*

SOLUCIÓN

a)

a.1) Descomposición en secuencia de microoperaciones u operaciones elementales:

1) **fetch**:

```
 i:    AR ← PC
       Z ← PC+1
 i+1:   ← M(AR) 1er ciclo de M
       PC ← Z
 i+2:  DR ← M(AR) 2o ciclo de M
 i+3:  IR ← DR
```

2) instrucciones:

```
1) LD .R1, #13[.R3]
       fetch
 i+4:  Z← IR.desp + R3
 i+5:  AR← Z
 i+6:   ← M(AR); 1er ciclo de M
 i+7:  DR ← M(AR); 2o ciclo de M
 i+8:  R1 ← DR;
2) ST .R3, #12[.R5]
       fetch
 i+4:  Z← IR.desp + R5
 i+5:  AR← Z
 i+6:  DR← R3
 i+7:   ← DR; 1er ciclo de M
 i+8:  M(AR) ← DR; 2o ciclo de M
3) ADD .R2, .R4, .R6
       fetch
 i+4:  Z ← R4 + R6; ACTUALIZAR_SR
 i+5:  R2 ← Z
4) POP .R7
        fetch
 i+4:  Z ← SP + 1; nuevo valor de SP
 i+5:  SP ← Z; actualiza SP
       AR ← Z
 i+6:   ← M(AR); 1er ciclo de M
 i+7:  DR ← M(AR); 2o ciclo de M
 i+8:  R7 ← DR;
5) RET
        fetch
 i+4:  Z ← SP + 1; nuevo valor de SP
 i+5:  SP ← Z; actualiza SP
       AR ← Z
 i+6:   ← M(AR); 1er ciclo de M
 i+7:  DR ← M(AR); 2o ciclo de M
 i+8:  PC ← DR;
```

a.2) En cada caso el número total de ciclos que tardaría en ejecutarse cada instrucción y su equivalente en tiempo es:

```
fetch: 4 ciclos, 120 ns
LD: 9 ciclos, 270 ns
ST: 9 ciclos, 270 ns
SUB: 6 ciclos, 180 ns
POP: 9 ciclos, 270 ns
RET: 9 ciclos, 270 ns
```

a.3) Una mejora inmediata y sustancial al *datapath* original del ejercicio sería la conexión directa del bus de datos al bus interno, lo que permitiría llevar los datos obtenidos de las lectura en memoria directamente a cualquier registro de la CPU y sin necesidad de pasar por el registro de datos, DR. Esta mejora produciría una reducción en el número de ciclos de todas las instrucciones al aprovecharse en el fetch, que es común a todas ellas.

Si se analiza la secuencia de microoperaciones obtenida para cada instrucción, se observa que ser obtendría una reduccción de al menos un ciclo, 30 ns, en todas ellas gracias al ciclo de menos en el fetch. Además, en el caso de la instrucciones que leen de memoria –LD, POP y RET– se ahorraría otro ciclo adicional al poderse hacer visible el contenido del DB directamente en el registro destino.

NOTAS: 20 de julio de 2018 **DURACIÓN: 1 h y 30 minutos**
REVISIÓN: 23 de julio de 2018 **PUNTUACIÓN: Especificada en cada ejercicio**