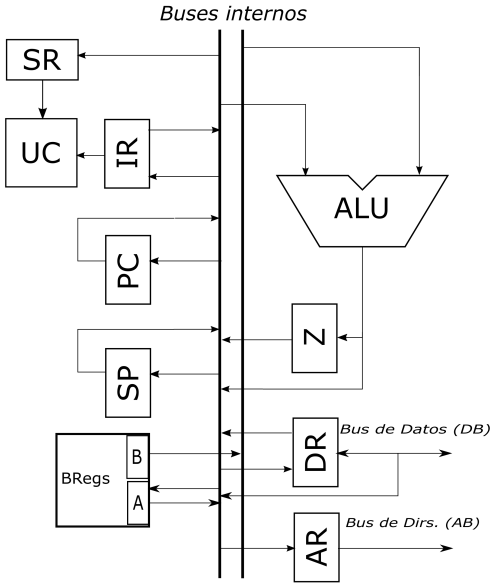
ESTRUCTURA DE COMPUTADORES. Grado en Matemáticas e Informática
SEGUNDO PARCIAL (14 de mayo de 2019)

---

1 (4 puntos) *Sea la CPU cuyo esquema simplificado (o* datapath*) 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 *o* temporal
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 del reg. de instrucción.*
4. *la memoria es direccionable a palabra y necesita para operar* tres *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 o microoperaciones para los casos que aparecen a continuación. Tenga en cuenta las siguientes indicaciones:*

- *en cada caso comience describiendo brevemente con palabras el funcionamiento o significado de cada una de las instrucciones propuestas, y de la fase de fetch que aparece primero.*
- *indique siempre claramente la o las microoperaciones* en cada uno de los ciclos de reloj *de la secuencia correspondiente.*
- *trate siempre de conseguir la secuencia con la menor duración posible.*

- *indique claramente con el texto ACTUALIZAR_SR los ciclos en los que se deba actualizar el registro de estado, SR.*

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

2) las siguientes instrucciones (señale con "fetch" la secuencia anterior, que se supone al principio de cada instrucción).

- *LD .R3, #13[.R6]*
- *SUB .R2, .R4, .R6*
- *CALL [.R6]*

a.2) *Según el 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.*

SOLUCIÓN

a)

a.1) Descomposición en secuencia de microoperaciones:

1) **fetch**:

Funcionamiento:

Lleva a IR la instrucción a la que apunta el PC y luego lo incrementa en una palabra:

"M(PC) → IR"

"PC+1 → PC"

Secuencia de microoperaciones:

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

2) instrucciones:

`1) LD .R3, #13[.R6]`

Funcionamiento:

Lleva al registro R3 el contenido de la dirección de memoria que corresponde al registro R6 más el desplazamiento:

"M(R6+IR.desp) → R3"

Secuencia de microoperaciones:

```
      fetch
i+4:  IR.desp + R6 → Z
i+5:  Z → AR
i+6:  M(AR) → ; 1er ciclo de M
i+7:  M(AR) → ; 2o ciclo de M
i+8:  M(AR) → R3; 3er ciclo de M
```

```
2) SUB .R2, .R4, .R6
```

Funcionamiento:

Resta al contenido de R4 al contenido de R6 y deja el resultado en R2, a la vez que actualiza el registro de estado, SR

"R4 - R6 → R2" y actualiza SR

Secuencia de microoperaciones:

```
       fetch
 i+4:  R4 - R6 → Z; ACTUALIZAR_SR
 i+5:  Z → R2
```

```
3) CALL [.R6]
```

Funcionamiento:

Apila el PC (que es la dirección de retorno) y pone en el PC el contenido de R6, que es la dirección de la rutina:

"Apilar PC"

"R6 → PC"

A su vez, apilar el PC conlleva las siguientes acciones:

"PC → M(SP)"

"SP - 1 → SP"

Secuencia de microoperaciones:

```
        fetch
 i+4:   SP → AR
        SP - 1 → Z; nuevo valor de SP
 i+5:   PC → DR
 i+6:   DR → M(AR); 1er ciclo de M
 i+7:   DR → M(AR); 2o ciclo de M
        Z → SP; actualiza SP
 i+8:   DR → M(AR); 3er ciclo de M
        R6 → PC; actualiza PC con el contenido de R6
```

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
SUB: 6 ciclos, 180 ns
CALL: 9 ciclos, 270 ns
```

2 (2 puntos) *Sea un computador con un ancho de palabra de 64 bits y cuyo sistema de memoria está compuesto por sólo dos niveles, memoria caché y memoria principal con las siguientes características:*

- *Memoria principal:*
  - *tiempo de acceso: 30 ns*
- *Memoria caché:*
  - *tiempo de acceso: 1 ns*
  - *tamaño de los bloques: 32 bytes*
  - *completamente asociativa*
  - *política de escritura: CBWA,* copy back with allocation

a) *Se pide calcular razonadamente los siguiente tiempos de acceso:*
a.1) *mínimo en lectura* ***a.2)*** *mínimo en escritura* ***a.3)*** *máximo en lectura* ***a.4)*** *máximo en escritura*

b) *Calcule el tiempo medio de acceso suponiendo una tasa de aciertos del 92 %. Razone si sería necesario conocer el porcentaje de lecturas, %R, y escrituras, %W. ¿Y la probabilidad de reemplazar un bloque que se encuentra modificado, %Modif? Si su respuesta es afirmativa en alguno de los dos casos, exprese su cálculo en función de estos porcentajes, %R, o %Modif, o de ambos.*

SOLUCIÓN

a)
a.1) mínimo en lectura El tiempo mínimo se corresponde con un acierto en Mca, y por tanto es 1 ns:

$$t_{LecturaMin} = t_{Mca} = 1\ ns$$

a.2) mínimo en escritura El tiempo mínimo en este caso se corresponde también con un acierto en Mca, y por tanto es 1 ns:

$$t_{EscrituraMin} = t_{Mca} = 1\ ns$$

a.3) máximo en lectura

El tiempo máximo corresponde al caso que produzca fallo en la caché y el bloque que se reemplaza esté modificado. En este caso habrá que copiar el bloque que se reemplaza en Mp ("bajarlo"), y luego copiar el bloque que produjo el fallo de la Mp a la caché (o "subirlo"). El tiempo de copiar un bloque a o desde memoria es de 120 ns, correspondientes a las cuatro palabras del bloque por los 30 ns de su tiempo de aceso:

$$t_{LecturaMax} = 1\ ns + 120\ ns + 120\ ns + 1\ ns = 242\ ns$$

a.4) máximo en escritura

En este caso es el mismo tiempo que el caso de las lecturas, puesto que en CBWA los tiempos implicados son los mismos, por lo tanto el valor que se solicita es 242 ns.

b)

En este caso, tanto las lecturas como las escrituras y los aciertos como los fallos pasan primero por la caché, y en el caso de los fallos –que ocurren en un porcentaje del 8 % para la tasa de aciertos dada– todos conllevan el tiempo de "subir" el bloque y un %Modif de ellos –expresado en tanto por uno– necesitará además el tiempo de "bajar" el bloque modificado:

$$\bar{t}_{acceso} = 1\ ns + 0,08 \times (\ \%Modif \times 120\ ns + 120\ ns + 1\ ns)$$

3 (4 puntos) *Se considera la representación compacta de matrices dispersas (matrices en las que la mayor parte de sus elementos son nulos) en forma de lista con los elementos no nulos, donde cada elemento se compone de tres campos enteros:*

- `elemento`*: Contiene el elemento que pertenece a la matriz.*
- `fila`*: Indica la fila a la que pertenece el elemento. Las filas comienzan a numerarse en la posición 0.*
- `columna`*: Indica la columna a la que pertenece el elemento. Las columnas comienzan a numerarse en la posición 0.*

El final de la lista está indicado con el el valor -1 contenido en el campo `fila`*. Los elementos de la lista no están ordenados por ningún criterio.*

Programe las siguientes subrutinas:

a) *La función* `agrega(p,valor,fila,columna)` *que añade el elemento descrito en los tres últimos parámetros a la matriz compacta indicada en* `p`*. Los parámetros se pasan en pila,* `p` *por dirección, los demás por valor.*

b) *La función* `elimina(p)` *que borra de la lista compacta* `p` *el primer elemento de dicha lista. El parámetro* `p` *se pasa en pila por dirección.*

Para la realización de este ejercicio 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.

Como ejemplo de la estructura de matriz compacta, se muestra en la figura la de una matriz en la que su elemento (0, 5) contiene el valor 15, el (1, 5) el valor 20 y el (4, 8) el valor 40. El resto de los elementos de la matriz son 0.

p	20	1	5	15	0	5	40	4	8	5	−1	−1

SOLUCIÓN

En ninguna de las dos subrutinas pedidas hay que llamar a otra subrutina ni hay que mantener variables locales, por lo que el manejo del marco de pila no es necesario. Si no se utiliza el regitro r1, tampoco sería necesario su salvaguarda para poder retornar correctamente.

a) La función agrega puede insertar el nuevo elemento al final de la lista. Para ello, busca el valor -1 en el campo fila de cada uno de los elementos de la lista que se pasa como primer parámetro. Cuando se encuentra, se inserta el elemento el sustitución del elemento final de la lista incluyendo el valor, fila y columna que se pasan como parámetros. A continuación del nuevo elemento se inserta un elemento finalizador de la lista que contiene el valor -1 en el campo fila. A continuación se muestra la función agrega:

```
agrega: ld r20,r30,r0        ; Inicia puntero a lista con parametro p
bucle:  ld r2,r20,4
        cmp r5,r2,-1
        bb1 eq, r5, fin_buc
        addu r20,r20,12      ; No se ha encontrado el final y
        br bucle             ; se avanza una posición en la lista.
fin_buc:ld r2,r30,4          ; Se carga y almacena valor
        st r2,r20,r0
        ld r2,r30,8          ; Se carga y almacena fila
        st r2,r20,4
        ld r2,r30,12         ; Se carga y almacena columna
        st r2,r20,8
        add r2,r0,-1         ; Se genera una última entrada en que la fila y
        st r2,r20,16         ; columna contienen -1
        st r2,r20,20
        jmp(r1)
```

b) La función elimina podría sobreescribir el último elemento de la lista compacta sobre el primero a eliminar, ya que no tiene que estar ordenada. Con esto, no sería necesario trasladar todos los elementos de la lista una posición hasta el principio de la lista. Se comprobaría primero que inicialmente la lista no está vacía. Luego, habría que recorrer la lista p buscando el elemento del final (cuya fila tiene el valor -1), y el elemento anterior a ése se sobreescribiría sobre el primero de la lista. Para recorrer la lista p hasta el final se utiliza el puntero en el registro r21, y en el registro r20 se tiene el puntero al primer elemento, tomándolo de los parámetros. El nuevo elemento finalizador de la lista será precisamente el anterior al último, y habrá que escribir el valor -1 en el campo fila para que pase a ser nuevo elemento finalizador de la lista.

```
elimina: ld r20,r30,r0     ; Inicia puntero a  principio de lista con parametro p
         or  r21,r20,r0    ; Inicializar puntero que recorre la lista hasta fin
         ld r2,r21,4       ; Si la fila del primer elemento de p es -1 (finalizador),
         cmp r5,r2,-1      ; está vacía y se salta a fin
         bb1 eq,r5,fin
         addu r21,r21,12   ; avanzar puntero a p
buc_fin: ld r2,r21,4       ; Si la fila del elemento de p es -1 (finalizador),
         cmp r5,r2,-1      ; se salta a copiar anterior al primer lugar
         bb1 eq,r5,primer
         addu r21,r21,12   ; avanzar puntero a p
         br buc_fin
primer:  ld r2,r21,-12     ; copiar valores del elemento anterior al finalizador en el primero
         st r2,r20,r0
         ld r2,r21,-8
         st r2,r20,4
         ld r2,r21,-4
         st r2,r20,8
```

```
        add r2,r0,-1      ; Se genera nuevo elemento finalizador con fila y
        st r2,r21,-8      ; columna a -1
        st r2,r21,-4
fin:    jmp(r1)
```

NOTAS: 28 de mayo **DURACIÓN: 1 hora y 50 minutos**
REVISIÓN: 30 de mayo **PUNTUACIÓN: especificada en cada ejercicio**