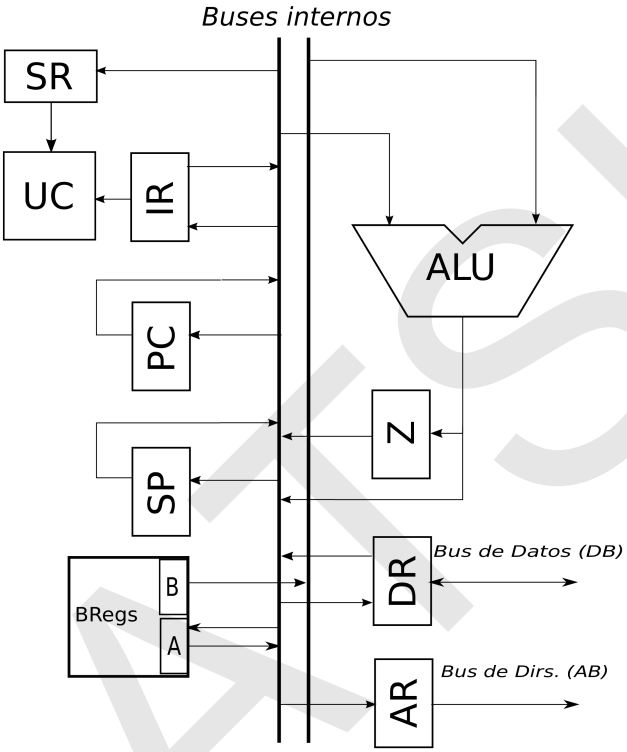
1 (5 puntos) *Sea la CPU cuyo esquema simplificado (o* datapath*) aparece en la figura. La ALU, todos los registros, rutas de datos y de direcciones son de 32 bits.*

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



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. *la memoria es direccionable a palabra y necesita para operar dos ciclos de reloj*
4. *el tiempo de ciclo de reloj es 50 ns*
5. *la pila crece hacia direcciones decrecientes y SP apunta a la primera dirección libre*

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

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

a.2) *las siguientes instrucciones (señale con "fetch" la secuencia anterior, que se supone al principio de cada instrucción). Indique claramente (con el texto ACTUALIZAR_SR) los ciclos en los que se debe actualizar el registro de estado, SR.*

1) ST .R3, #12[.R5]

2) SUB .R2, .R4, .R6

3) PUSH .R7

4) RET

b) *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.*

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

SOLUCIÓN

a)

a.1) Descomposición en secuencia de microoperaciones:

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
```

a.2) Instrucciones:

```
1) 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
2) SUB .R2, .R4, .R6
          fetch
    i+4:  Z ← R4 - R6; ACTUALIZAR_SR
    i+5:  R2 ← Z
3) PUSH .R7
           fetch
    i+4:  AR ← SP
          Z ← SP - 1; nuevo valor de SP
    i+5:  DR ← R7
    i+6:   ← DR; 1er ciclo de M
          SP ← Z; actualiza SP
    i+7:  M(AR) ← DR; 2o ciclo de M
4) 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;
```

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

```
fetch: 4 ciclos, 200 ns
ST: 9 ciclos, 450 ns
SUB: 6 ciclos, 300 ns
PUSH: 8 ciclos, 400 ns
RET: 9 ciclos, 450 ns
```

c) Una mejora inmediata, que permitiría ahorra un ciclo en todas lecturas de memoria, sería que hubiese un camino directo desde el bus de datos hasta el interior de la CPU y evitando el registro de datos, DR. Este cambio significaría un ciclo menos en el fetch de todas las instrucciones y otro ciclo por cada acceso a memoria en lectura para las instrucciones que los realizaran.

Otra mejora posible se puede detectar en las operaciones elementales correspondientes a las instrucciones 1) y 4), donde el resultado de una operación de la ALU se debe llevar al registro AR, y donde se evitaría un ciclo si existiese una conexión directa entre esta salida y el citado registro, lo que se podría hacer con un multiplexor a la entrada del registro, o con un posible bus de direcciones interno.

Además, si los registros PC y SP dispusieran ambos de la lógica necesaria para autoincrementarse y/o autodecrementarse sin necesidad de usar la ALU se podría reducir algún ciclo. .

2 (5 puntos) *Un computador cuenta con dos formatos de representación de números:*

- *Formato 1: Coma fija, 12 bits en complemento a uno con la coma entre los bits 5 y 6, por ejemplo: 100001,011101*
- *Formato 2: Coma flotante, 12 bits. El bit superior representa el signo, los cinco siguientes el exponente representado en exceso a 16 y los 6 siguientes la magnitud de la mantisa, representada en signo-magnitud con bit implícito y la coma a la derecha de éste.*

a) *Determine el rango y resolución de ambos formatos.*

b) *La cadena H'DD6 representa al número A en el formato 1. Determine su valor decimal y represéntelo en el formato 2.*

c) *La cadena H'B87 representa al número B en el formato 2. Determine su valor decimal y represéntelo en el formato 1.*

d) *Realice la operación A-B en ambos formatos, dejando el resultado en el formato de partida. Determine el valor decimal del resultado obtenido. Para el formato 2 utilice dos bits de guarda y un bit retenedor.*

e) *Rediseñe el formato 2 (manteniendo el número total de bits) para que el máximo número representable del formato 1 se pueda representar en el formato 2 con una resolución que sea 8 veces la del formato 1.*

SOLUCIÓN

a) Rango y resolución de los formatos.

Formato 1.

$$\begin{cases} \texttt{100000,000000} = \texttt{-01111,111111} & \rightarrow & -(2^5 - 2^{-6}) \\ \texttt{011111,111111} & \rightarrow & 2^5 - 2^{-6} \end{cases}$$

$Rango = [-(2^5 - 2^{-6}), 2^5 - 2^{-6}]$

$Resolucion = 2^{-6}$

Formato 2.

Exponente: $[-16, 15]$. Como hay que reservar el exponente -16 para la representación del cero, el rango del exponente queda: $[-15, 15]$.

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

$Rango = \pm\left[1 \cdot 2^{-15}, (2 - 2^{-6}) \cdot 2^{15}\right] \cup 0$

$Resolucion = 2^{-6} \cdot 2^E$

b) Valor decimal y representación en el formato 2.

$A = \texttt{H'DD6} = \texttt{110111,010110} = \texttt{-001000,101001} = \texttt{-8,640625}$

$A = \texttt{-1,000101} \cdot 2^3 = \texttt{1 10011 000101} = \texttt{H'CC5}$

c) Valor decimal y representación en el formato 1.

$B = \texttt{H'B87} = \texttt{1 01110 000111} = \texttt{-1,000111} \cdot 2^{-2} = \texttt{-0,27734375}$

$B = \texttt{-000000,010001} = \texttt{111111,101110} = \texttt{H'FEE}$

d) $A - B$**.**

Formato 1.

$A - B = A + (-B)$

A	110111,010110
$-B$	000000,010001
$A + (-B)$	110111,100111

Como no hay acarreo más allá del bit más significativo no hay que realizar la realimentación de acarreo.

$A + B = \texttt{H'DE7} = \texttt{-001000,011000} = \texttt{-8,375}$

Formato 2.

$A - B = A + (-B)$

Los números a sumar son: $A = \texttt{-1,000101} \cdot 2^3 \quad y \quad -B = \texttt{+1,000111} \cdot 2^{-2}$

La diferecia de exponentes: $E_A - E_B = 3 - (-2) = 5$ nos dice que hay que desplazar la mantisa de -B cinco lugares a la derecha. Al desplazar cinco lugares (teniendo en cuenta los dos bits de guarda y el bit retenedor) queda:

$-B = \texttt{+0,000010 00 1} \cdot 2^3$

Se realiza la suma de mantisas (resta):

M_A	$-$ 1,000101 00 0	
$-M_B$	$+$ 0,000010 00 1	
$A + (-B)$	$-$ 1,000010 11 1 $\cdot 2^3$	*Esta normalizado*
	$+$ 1	*Redondeo*
$A + (-B)$	$-$ 1,000011 01 1 $\cdot 2^3$	

$A - B = \texttt{-1,000011} \cdot 2^3 = \texttt{1 10011 000011} = \texttt{H'CC3}$

$A - B = \texttt{-1000,011} = \texttt{-8,375}$

e) Rediseño del formato.

El máximo número representable en el formato 1 es `011111,111111`, que representado en el formato 2 sería $\texttt{-1,111111} \cdot 2^4$ es decir, que se representaría en coma flotante con exponente 4

En coma fija el número se representa con resolución 2^{-6}. Como nos piden representarlo con una resolución 8 veces ésta, será $2^{-6} \cdot 2^3 = 2^{-3}$

Si p es el número de bits de la mantisa, la resolución del formato de coma flotante es $2^{-p} \cdot 2^E$. Así, se debe cumplir:

$2^{-p} \cdot 2^4 = 2^{-3} \rightarrow p = 7$

El formato quedaría con un bit para el signo, cuatro para el exponente y siete para la mantisa. Vemos que con cuatro bits el rango del exponente sería $[-7, 7]$ con lo que se sigue pudiendo representar el máximo valor del formato 1.