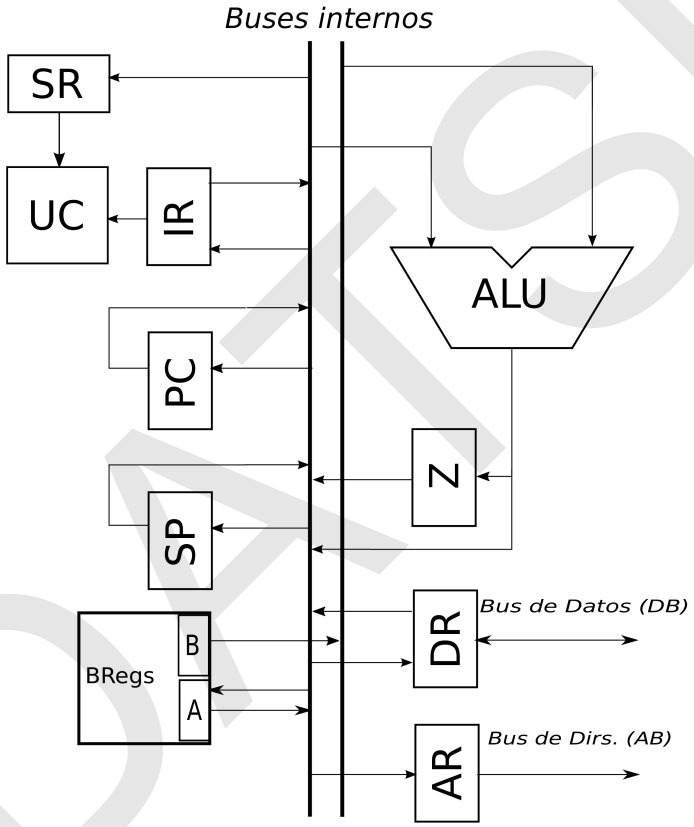
ESTRUCTURA DE COMPUTADORES (GradoII)
EXAMEN FINAL DE JULIO. SEGUNDO PARCIAL (4 de julio de 2018)

---

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



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 o microoperaciones, 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) LD .R1, #13[.R3]

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

3) ADD .R2, .R4, .R6

4) POP .R7

5) RET
```

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 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 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 reducción 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.

2 (5 puntos). *Se tiene un formato de coma flotante de 16 bits cuyo bit superior representa el bit de signo, los cinco siguientes el exponente expresado en exceso a 16 y los diez bits siguientes la magnitud de la mantisa, normalizada con bit implícito y con la coma a la izquierda de dicho bit.*

a) *Determine el rango de representación del formato especificando cómo representaría el cero.*

b) *$A = H'3CB5$ es una representación en este formato. Determine su valor decimal.*

c) *Represente en el formato el número decimal $B = -50,0125$.*

d) *Realice la operación $A+B$ utilizando 2 bits de guarda, bit retenedor y redondeo al más próximo, dejando el resultado en el formato de almacenamiento y determine su valor decimal.*

e) *Rediseñe el formato de representación sin variar el número total de bits para que se pueda representar el número 10^{10}.*

SOLUCIÓN

1. **Rango y resolución del formato.**

   Exponente: Como está representado en exceso a 16 su rango es $[-16, 15]$. Como el formato tiene bit implícito, hay que reservar un valor del exponente para la representación del cero. Reservamos el valor -16 (todo ceros). Por lo tanto, el rango del exponente es $[-15, 15]$. El cero se representará con los 16 bits a cero.

   Mantisa: Como está representada en signo magnitud su rango es simétrico, siendo:

Mantisas positivas: $\begin{cases} \texttt{,10000000000} & \rightarrow & 2^{-1} \\ \texttt{,11111111111} & \rightarrow & 1-2^{-11} \end{cases}$

El rango es: $\pm\left[2^{-1}\cdot 2^{-15}, (1-2^{-11})\cdot 2^{15}\right] \cup 0$

La resolución es: $2^{-11}\cdot 2^{E}$

2. **Valor decimal de un número.**

   $A = \texttt{H'3CB5} = \texttt{0011 1100 1011 0101} = \texttt{0 01111 0010110101} = \texttt{+,10010110101}\cdot 2^{-1}$

   $A = \texttt{+,010010110101} = \texttt{+0,294189453}$

3. **Representación de un número decimal.**

   $B = -50,0125 = \texttt{-110010,00000011} = \texttt{-,11001000000}\cdot 2^{6}$

   Quedando en el formato $B = \texttt{1 10110 1001000000} = \texttt{H'DA40}$.

4. **Suma $A+B$.** Los números a sumar son:

   $A = \texttt{+,10010110101}\cdot 2^{-1}$ $\quad y \quad$ $B = \texttt{-,11001000000}\cdot 2^{6}$

   Determinamos la diferencia de exponentes: $E_A - E_B = -1-6 = -7$. Hay que desplazar la mantisa de A siete lugares a la derecha.

   Suma de las mantisas (resta):

B	$-,\texttt{11001000000 000}\cdot 2^{6}$	
A	$+,\texttt{00000001001 011}\cdot 2^{6}$	
$A+B$	$-,\texttt{11000110110 101}\cdot 2^{6}$	Normalizado
	$+,\texttt{00000000000 100}$	Redondeo
	$-,\texttt{11000110111 001}$	$\cdot 2^{6}$

   $A+B = \texttt{-,11000110111}\cdot 2^{6} = \texttt{1 10110 1000110111} = \texttt{H'DA37}$

   $A+B = \texttt{-110001,10111} = \texttt{-49,71875}$

5. **Rediseño del formato.**

   En este formato el número más grande representable es $(1-2^{-11})\cdot 2^{15}$. En general, llamando p al número de bits de la mantisa y E_{max} al máximo exponente, tendremos que dicho número sería $(1-2^{-p})\cdot 2^{E_{max}}$.

   Así, se debe cumplir: $(1-2^{-p})\cdot 2^{E_{max}} > 10^{10}$

   Como 2^{-p} es despreciable frente a la unidad podemos aproximar por: $2^{E_{max}} > 10^{10}$

   Tomando logaritmos: $E_{max}\cdot log2 > 10 \rightarrow E_{max} > 33,21$

   Necesitamos representar los exponentes en exceso a 64, es decir, con 7 bits. Así pues, el formato queda con un bit para el signo, siete para el exponente y ocho para la mantisa.

NOTAS: 20 de julio de 2018 **DURACIÓN: 1 hora y 25 minutos**
REVISIÓN: 23 de julio de 2018 **PUNTUACIÓN: Especificada en cada apartado**