Tema 2

Teoria

Procesador

Introducción

Recordar que hace la CPU , Ejecuta instrucciones (función básica) , Resuelve situaciones anómalas (desbordamiento, operación no válida, error de paridad, etc) y Controla la comunicación con periféricos.
Reloj: tren de pulsos caracterizado por su periodo

CAMINO CRÍTICO: camino de máximo retardo entre un origen y un destino. Depende de los dispositivos que tengan que atravesar las señales.

Lectura
1. Dirección -> AR
2. M(AR) -> Reg; lectura

Escritura
1. dirección -> AR
2. dato DR
3. DR M(AR) ; escritura

UC Cableada

Se construye mediante métodos generales del diseño lógico, más rápidas, diseño y construcción complejo y costoso. Las soluciones actuales permiten construir máquinas rápidas.

UC Microprogramada

Utiliza la memoria para almacenar el estado de las señales de control en cada periodo de las ejecuciones de cada instrucción.

UC CABLEADA	UC MICROPROGRAMADA
Ventajas •Alta velocidad de funcionamiento •Implementaciones más pequeñas (reducido número de componentes)	Ventajas •Son sistemáticas con un formato bien definido •Pueden ser fácilmente modificables durante el proceso de diseño
Inconvenientes •Difícil realizar modificaciones en el diseño: modificación → rediseño •No tienen estructura común	Inconvenientes •Requieren más componentes para su implementación •Tienden a ser más lentas que las unidades de control cableadas debido a que tienen que realizar operaciones de lectura de una memoria para obtener las señales de control
▪Técnica de diseño favorita en las arquitecturas RISC	▪Emulación •Se puede utilizar un microprograma para que la UC interprete otro lenguaje máquina distinto (una máquina a emular) sin necesidad de realizar modificaciones en el hardware de la unidad de control -> cambiando sólo el microprograma!

Microinstrucción

Cadena de ceros y unos que representan la activación o no del conjunto de señales. Cada palabra de la memoria de control es una microinstrucción.

Microprograma

Secuencia de $\mu I'S$ cuya ejecución permite interpretar una instrucción.

Memoria de Control

Memoria que almacena los microprogramas o a partir de los que se obtienen las señales de control necesarias para la ejecución del repertorio.

Excepciones

Suele ser cuando ocurre un overflow

1. Salvar el estado de la máquina PC y la actual pila.
2. Pasar a modo supervisor
3. Saltar a rutina de SO que maneje las excepciones
4. Retornar al programa

Internas -> Excepciones
Externas -> Comunicación con el periférico.
TRAP - Llamada al sistema.

Interrupción

Evento externo inesperado rompe la ejecución del programa. Causadas por eventos externos. Asincorna a la ejecución del programa

Aritmetica

Terminologia

- Rango de representación: Intervalo entre el mayor y el menor número representables.
- Resolución: Diferencia entre los valores de un número representable y el inmediato siguiente.
- Desbordamiento Cuando un resultado está fuera del rango de representación.

Complemento A2

- Representación :
  - N=6 B=101110
  - A = 7
  - A = 000111
  - -A => 100000 - 000111 = 111001 = 111000+1
  - |B|= 1000 000 - 101110 = 010010 = 18
  - Valor maximo = 011 111= $2^5 - 1 = 31$
  - Valor mínimo = 100 000= $-2^5 = -32$
- Rango = $[-2^{n-1}, -1] \cup [0, 2^{n-1} - 1]$
  - Resolución = 1

Complemento A1

  - N=6 B=101110
  - A = 7
  - A = 000111
  - -A => 111 111 - 000111 = 111001 = 111 000
  - |B|= 111 111 - 101110 = 010001 = 17
  - Valor maximo = 011 111= $2^5 - 1 = 31$
  - Valor mínimo = 100 000= -011 111= $-(2^5 - 1) = -32$
- Rango = $[-(2^{n-1} - 1), -1] \cup [0, 2^{n-1} - 1]$
  - Resolución 1

Signo-Magnitud

  - N=6 B=101110 A = 7
  - A = 000111 -A= 100111 B = -14 -B=001110

- Rango y resolución igual que en complemento a 1
- Suma A+B = $(-1)^s \times M$ siendo A= $(-1)^{sA} \times MA$ y B= $(-1)^{sB} \times MB$ y utilizando un sumador en binario sin signo:
    - Si SA=SB ir a 5
    - Si MA<MB ir a 4
    - S=SA, M=MA-MB,FIN
    - S=SB, M=MB-MA,FIN
    - S=SA=SB, M=MA+MB, si CY= 1 hay OVF, FIN

Entero en Exceso a "M"

- M = 2^{n-1} o M= $2^{n-1} - 1$ (el usado en el estándar IEEE)
- N=6 M=32 B=001 110 A = 7
- A = 7+32 =39 = 100111 B = 001 110 - 32= 14-32=-18
    - Valor maximo = 111 111= $63 - 32 = 31$
    - Valor mínimo = 000 000= $0 - 32 = -32$
- Rango = $[M, -1] \cup [0, 2^{n-1} - 1 - M]$ Resolucón = 1

Coma Flotante

- V(X)= $M \cdot r^E$
    - M = Mantisa o Fracción
    - R = Base o Radix
    - E = Exponente o Característica (q bits)
- Normalmente r = 2^K
- Mantisa: coma fija con signo y base r
- Exponente: Entero y base 2
- Rango exponente = [-64,63]
    -

S 1	Exponente 7	Mantisa 8

    - Rango mantisa :
        - ,0000 0000 = 0
        - ,0000 0001 = 2^8
        - ……
        - ,1111 1111 = $1 - 2^{-8}$
        - $\pm [2^{-8} \cdot 2^{-64}, (1 - 2^{-8}) \cdot 2^{63}] \cup 0$
    - Resolución = $2^{-8} \cdot 2^E$
- Normalización : Un número en coma flotante está con su mantisa normalizada si al desplazar la mantisa un dígito a la izquierda y decrementar el exponente en 1 cambia el valor del número
    -

S 1	Exponente 7	Mantisa 8

- Rango Mantisa
  - ,1000 0000 = 2^{-1}
  - ……
  - ,1111 1111 = $1-2^{-8}$
  - $\pm[2^{-1}\cdot 2^{-64},(1-2^{-8})\cdot 2^{63}]$
  - Problemas de normalización , resultado de operaciones no normalizadas, el cero no es representable
- Bit implícito: A un número en coma flotante con r=2 y su mantisa en signo magnitud y normalizada, puede dejarse el bit más significativo como implícito ya que tiene que ser un 1.
  -

S 1	Exponente 7	Mantisa 8

  - Rango Mantisa
    - ,1 0000 0000 = 2^{-1}
    - ……
    - ,1 1111 1111 = $1-2^{-9}$
    - $\pm[2^{-1}\cdot 2^{-64},(1-2^{-9})\cdot 2^{63}]$
- Suma y Resta
  - Pasos
    - Compara exponentes
      - Identificar mantisa a desplazar
      - Determinar el número de desplazamiento
      - Utilizar el restador
    - Desplazar Mantisa de exponentes menor
    - Sumar/Restar mantisas
      - Depende de la representación de las mantisas
      - Depende del operador que se utilice
      - Puede haber OVF -> Hay que normalizar
      - La resta no es conmutativa
    - Detectar resultado cero
      - Se detecta con el flag, Z=1
      - Se devuelve la representación definida para el 0
    - Normalizar
      - OVF=1 despl dcha. M` y R <- E+1
      - OVF=0 despl izqda. M` y R <- E+x
      - N=1,0,-1,...,-(p-1)= cantidad a sumar el exponente mayor
    - Corregir exponente
      - Seleccionar el exponente mayor
      - Sumar N
      - Sumar 1 si hay post normalización tras el redondeo
    - Detectar desbordamiento
      - Si E > Exponente mayor, overflow
      - Si E < Exponente menor, underflow

Redondeo

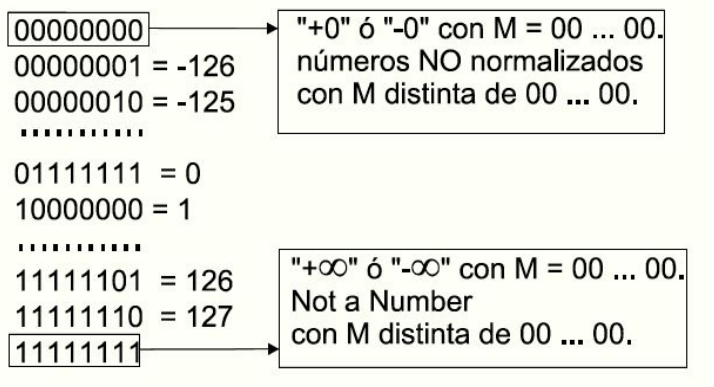
- Truncamiento : suprimir bit sobrantes
- Forzado a 1: truncamiento dejando siempre a 1 el bit menos significativo
- Redondeo al más príximo
- Redondeo a cero, a $+\infty$ y $a\ -\infty$

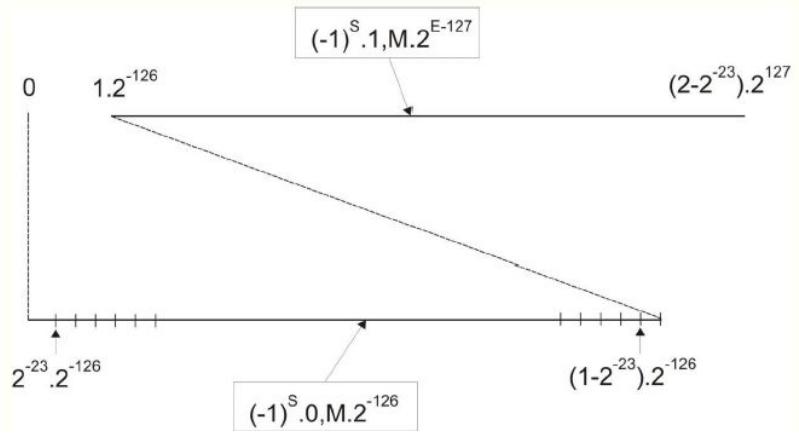
Dígito de Guarda y Bit retenedor

- Dígito de guarda: dígitos añadidos a la mantisa para obtener la precisión máxima.En el caso signo-magnitud se necesitan dos bits de guarda, uno para normalizar y otro redondeo.
- Bit retenedor : bit que se añade para propagar el borrow en la resta.

Estándar IEEE 754 de coma flotante

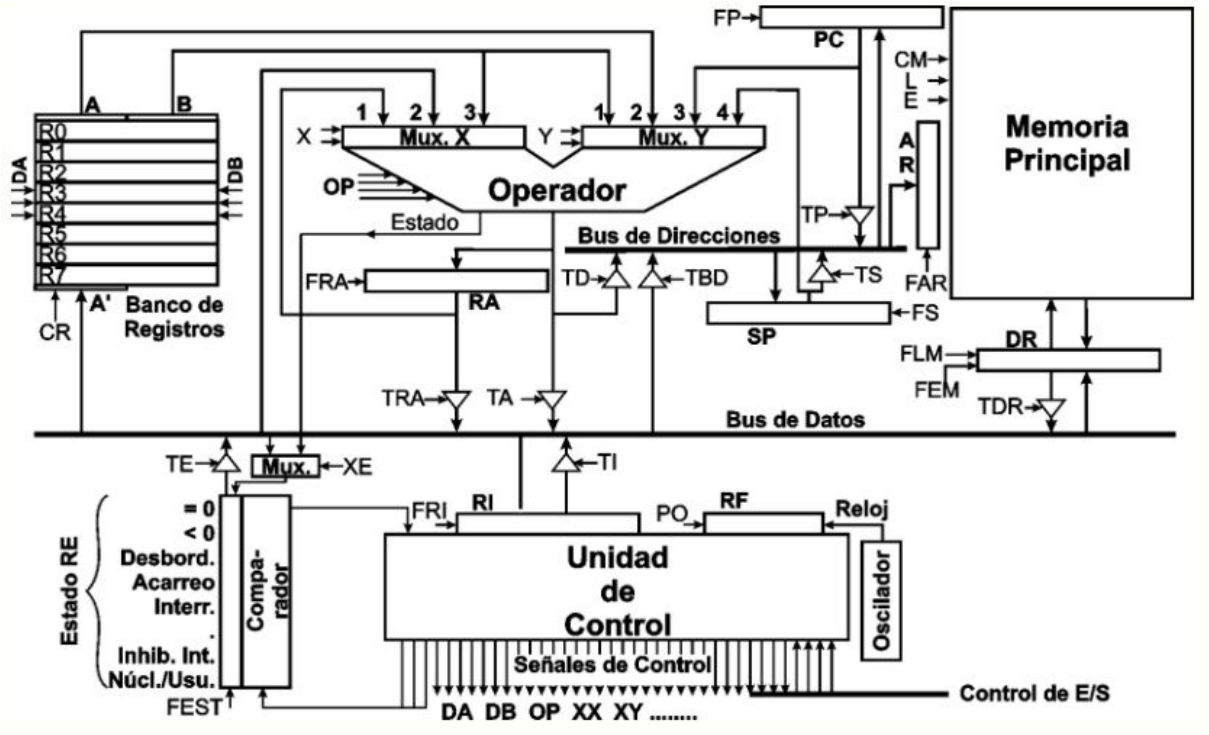
S 1	Exponente 8	Mantisa 23





Ejercicios de clase

Procesador

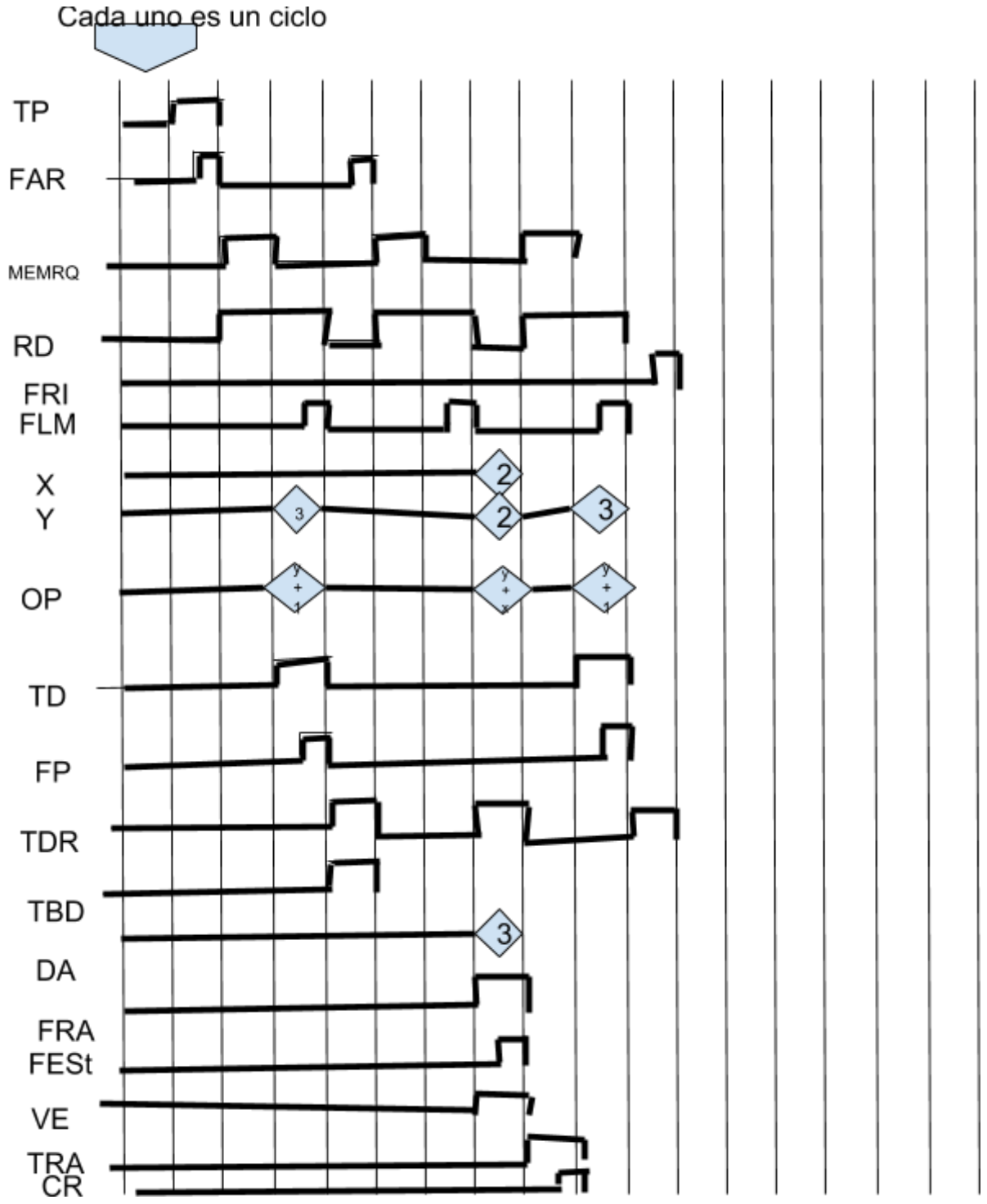


ADD .3,/100

1. ¿Que hace la instrucción?
    a. Tiene 2 palabras
    b. Toma lo que hay en mil y lo carga en el registro r3 sumando M[1000]+ r3
    c. Tengo lectura por lo que se que la dir va a AR , leer
2. Desglosar en operaciones elementales mirando los caminos posibles por el procesador
    a. PC ->AR
    b. M(AR) -> DR
        i. Hemos leído la 2 palabra /1000
    c. DR -> AR
    d. M(AR)-> DR
        i. Hemos leído el contenido de M[/1000]
    e. RA <- R3 + DR, act RE
        i. Va a RA para evitar conflictos en el bus de datos
    f. R3 <- RA
        i. Fin fase de operación
        ii. Pasamos a realizar la fase de fetch de la siguiente operación
    g. Pc -> AR

h. M(AR)-> DR
i. PC+1 -> Pc
j. DR -> RI

3. Solapar operaciones elementales
    a. PC ->AR
    b. M(AR) -> DR ; PC+1 -> PC (2º ciclo)
    c. DR -> AR
    d. M(AR)-> DR
    e. RA <- R3 + DR, Pc -> AR
    f. R3 <- RA, M(AR)-> DR
    g. PC+1 -> Pc
    h. DR -> RI
4. Cronograma



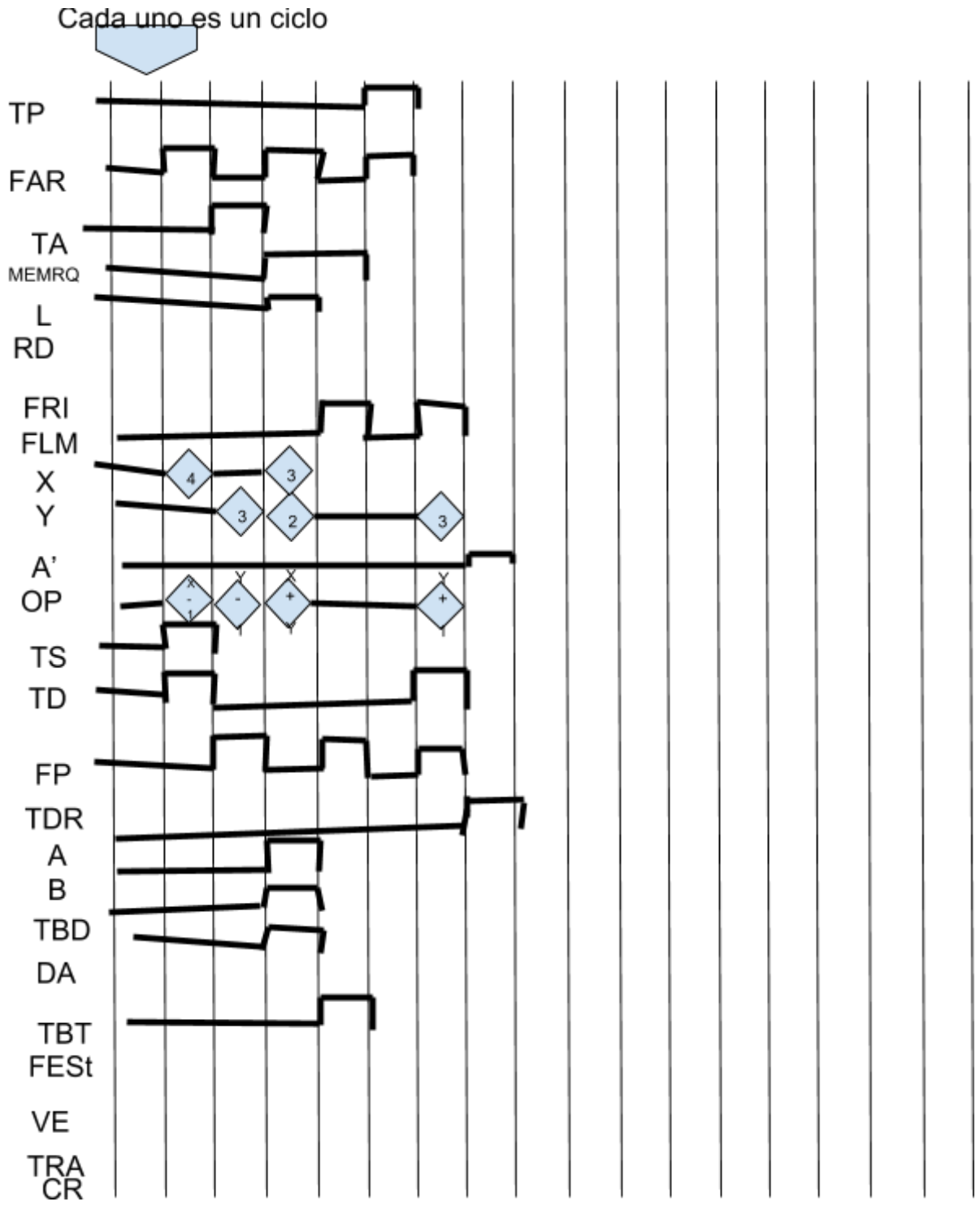
CALLZ [#7 [.R5]]

1. ¿Que hace la instrucción?
    a. Es un salto con retorno
    b. Salvar DR= Push(PC)
    c. PC <- X; PC <- M(R%+7)
        i. Si Z . Fetch

2. Desglosar en operaciones elementales mirando los caminos posibles por el procesador
    a. SP -1 ->SP
    b. SP -> AR
    c. PC -> DR
    d. DR -> M(AR)
    e. BR(R5) + RI(desp)->AR
    f. M(AR)->DR
    g. DR -> PC
    h. PC -> AR
    i. M(AR)-> DR; PC ++
    j. DR -> RI

3. Solapar operaciones elementales
    a. SP -1 ->SP, AR
    b. PC -> DR
    c. DR -> M(AR), BR(R5) + RI(desp)->AR
    d. M(AR)->DR, PC
    e. PC -> AR
    f. M(AR)-> DR; PC ++
    g. DR -> RI
4. Cronograma

    a. SP -1 ->SP, AR (4,Td,TS,FAR)
    b. PC -> DR (FP,3,TA)
    c. DR -> M(AR), BR(R5) + RI(desp)->AR (L,A,B,1,2,TA,TBD,FAR)
    d. M(AR)->DR, PC (FLM,TBT,FP)
    e. PC -> AR (TP,FAR)
    f. M(AR)-> DR; PC ++ (FLM,3,TD,FP)
    g. DR -> RI (TDR,A')


Cada uno es un ciclo
TP
FAR
TA
MEMRQ
L
RD
FRI
FLM
X
Y
A'
OP
TS
TD
FP
TDR
A
B
TBD
DA
TBT
FESt
VE
TRA
CR
4
3
3
2
3
+
+

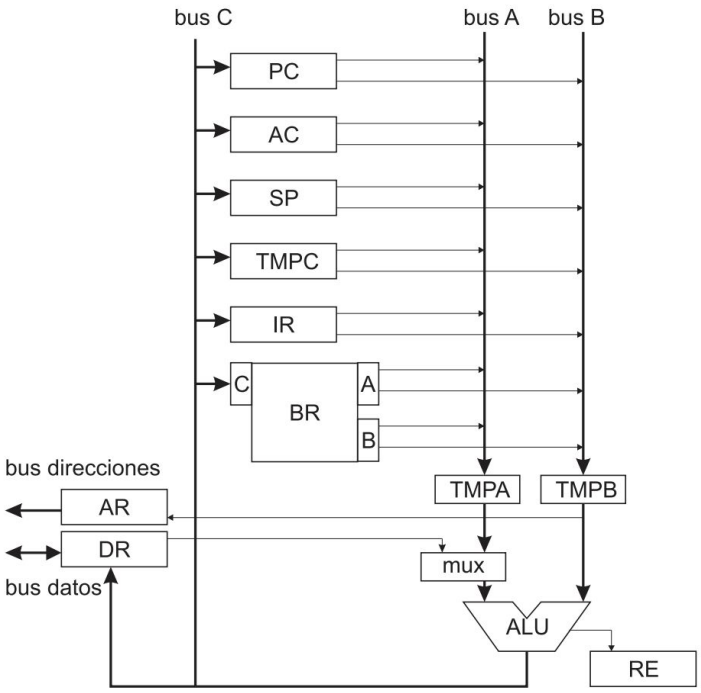


Figura 1. Estructura de la CPU

8 Sea la estructura del procesador de la figura 1. A través de los buses de datos y de direcciones se conecta con una memoria as´ıncrona que activa una señal READY cuando ha acabado la operación solicitada y cuyo tiempo medio de acceso es de 110 ut.

Puntos a buscar
-> Reg Temporal
->dir a bit/pal
->Tall -> nº ciclos de reloj
-Sincrona
-Asincrona
->oper ALU

a) Añada en la figura las se˜nales de control que estime necesarias.

Por cada registro una señal de carga de registro (Fpc , FAC, FSP, FTMPC).
Todos los accesos a un bus deberían estar con un triestado , y una señal activa por nivel.

b) Calcule el tiempo de ciclo de reloj para este procesador en el caso de que la unidad de control sea microprogramada. Los retardos de algunos dispositivos del computador con: Acceso al banco de registros (BR): 8 ut Acceso a registro: 4 ut Multiplexores: 1 ut Triestado: 1 ut Operaci´on ALU: 54 ut Acceso a Memoria de Control: 38 ut Decodificador: 5 ut

$$TCK = TRC + TR\hat{} + TMA + TALV + \{TRE/TBR + TRE + 2Tmux + TDEC + TDP + 2TDP + TMC + T$$
$$4 + 4 + 1 + 54 + 4 * 2 + 2 * 1 + 5 + 2 * 4 + 38 + 4 = 128$$

Tacc - Necesita 2 ciclos de reloj

c) ¿Qué modos de direccionamiento podrá admitir esta estructura del procesador? ¿Sería posible el direccionamiento relativo a registro base? Proponga algún cambio para mejorar el rendimiento con este modo de direccionamiento y explique cómo se lograr´ıa dicha mejora.

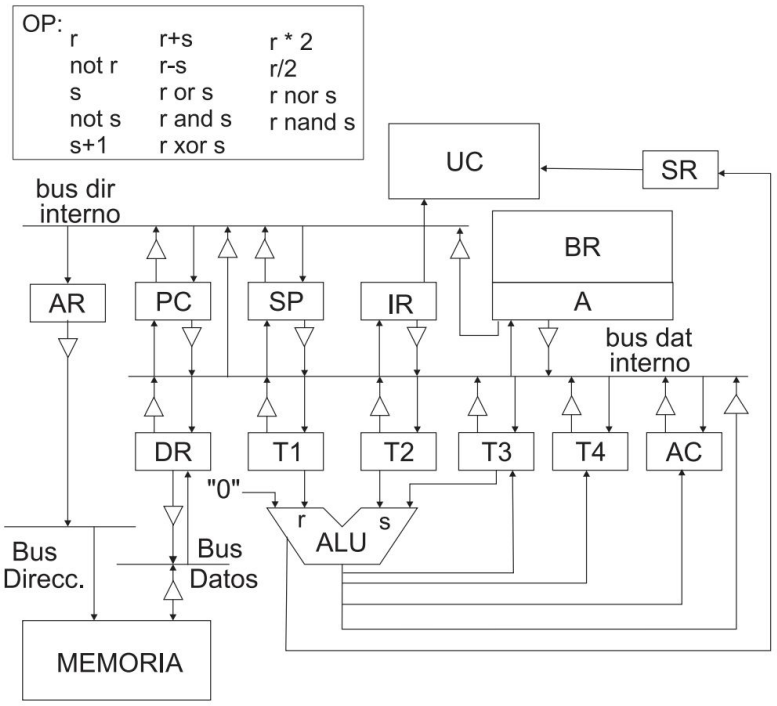
#desp[Rb]
RI desp -> TMPA; BR(RB) ->TMPB
TMA + TMPB -> TMPC
TMPC -> TMPB
TMPB -> AR

FETCH
f1 PC -> TMPB
f2 TMPB -> AR; TMPB + 1 -> PC
f3 M(AR)-> DR; Si RD4 ir f3
f4 DR -> RI, ir a C.O

OR .R1,R2,/dir
O1 Pc->TMPB
O2 TMPB -> AR; TMPB + 1 -> PC
O3 M(AR)->DR; Si RD4, ir a o3
O4 DR+O -> TMPC
O5 TMPC -> TMPB
O6 TMPB -> AR
O7 M(AR)->DR; SI RD4 IR A O7
BR(R2)->TMPB
O8 DR , TMPB -> BR(R1); RE

Tacc = 110
Tejec = (4+1)+ (8+2) = 15
T = 15*128=192

Ejercicio 17



Problemas de Clase

Problema 10 de clase Artim

Estandar IEEE

S 1	EXP 8 $2^{n-1}-1$	Mantisa 23

Exceso 127
[-127,127]

Normalización : en coma flotante se puede representar el número de diferentes maneras.
0000 0000 -> 0
0……..1
11…….0
1111 1111 -> ∞
No normalizados
Exp minimo = -126 ya que el -127 está usado para el 0
0,0……...1 $\times 2^{-126}$
0,1……...1 $\times 1-2^{-126}$

Problema 9

S **1**	Exp **7**	Mantisa **8**

Rango y Resolución de formato

Normalizados

Esp 2^{n-1} exceso 63

Exp [-62,63]

Mantisa $1,0000\ 0000$ -> 1

$1,1111\ 1111 \rightarrow 2 * 2^{-8}$

$[1 * 2^{-62},\ (2 - 2^{-8}) * 2^{63}]$

No normalizados

Exp = -62

Mantisa $+0,0000\ 0001 \rightarrow 2^{-8}$

$+0,1111\ 1111 \rightarrow 1 - 2^{-8}$

$[2^{-8} * 2^{-62}, (1 - 2^{-8}) * 2^{-62}]$

$$[1 * 2^{-62},\ (2 - 2^{-8}) * 2^{63}] \cup 0 \cup [2^{-8} * 2^{-62}, (1 - 2^{-8}) * 2^{-62}]$$

Resolución -> $2^{-8} * 2^{E}$

2)

a) +0,1100 1011 111 $\times 2^{-3}$
   i) Normalizarlo
   ii) 1,1011 0111 110 $\times 2^{-4}$
   iii) -1
   iv) 1,1011 1000 0 $\times 2^{-4}$
   v) 0 011 1011 1011 1000 = H´3BB8

b) -0,1111 0111 101 $\times 2^{10}$
   i) -1,1110 1111 010 $\times 2^{9}$
   ii) 1,1110 1111
   iii) 1 100 1000 1110 1111 = H´C8EF

c) -0,0111 1010 101 $\times 2^{15}$
   i) -1,1110 1010 100 - $\times 2^{13}$
   ii) Norm
   iii) 1,1110 1010 $\times 2^{13}$
   iv) 1 100 1100 1110 1010 = H´CCEA

d) 0,0000 0110 110 $\times 2^{-59}$

i) 1,1011 0000 000 $\times 2^{-65}$ OVF

ii) + 0,0011 0110 000 $\times 2^{-62}$

A

0	100 0101	1100 0101

= + 1,11000101 $\times 2^{6}$ = + 11110001,01= 113,25 (7 * 16 + 1*16 = 113)

(si exp 100 0101 el primer 1 se toma como el signo , y con el resto se calcular 1+4= 5 y le sumamos 1)

B

1	011 1111	0010 1001

= - 1,00101001 $\times 2^{0}$ = -1,16015625

011 1111 ($2^{6} - 1 = 63 - exceso\ 63$)

A+B

Ma = + 1,11000101 00 0 $\times 2^{6}$

Mb = - 0,0000 0100 10 1 $\times 2^{6}$

1,1100 0000 01 1 $\times 2^{6}$ -> 112

$$Error\ Absoluto\ |112 - 112,0898438| = 0,00858438$$

Ejercicio 14-15

Coma implicita

Si tiene bit implícito esta normalizado

S 1	Exp 8	MANT 15

a) Rango y Resolución

i) Exp [-127,127]

ii) MAnt [,10….0 -> 2^{-1}] [1,1…...1 -> 1- 2^{-16}] (recordar 10 por coma implicita)

b) Num decimales

i) A = -30,84

1) Parte entera entre 16 -> 1 14 13 7 0

2) Pasar a binario 0001 1110 1101 0111 0000

3) -0,1111 0110 1011 $\times 2^{5}$

4) H`C2F6B9

ii) B = 0,3244

1)

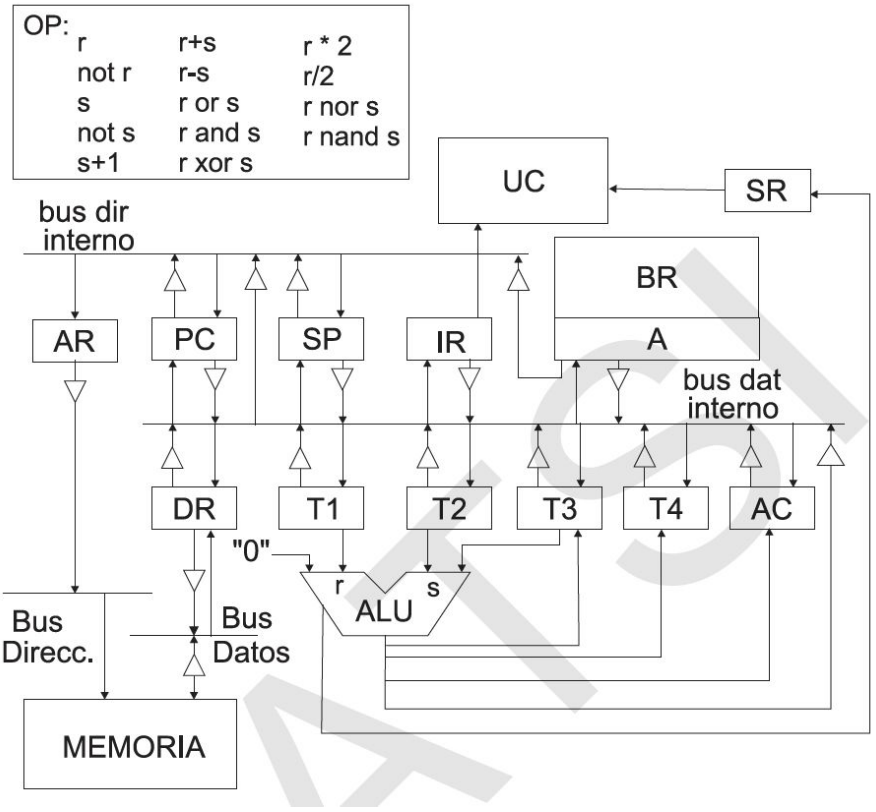
c) A+B

i) 2bq

ii) 1 br

Noviembre 2018

1 (5 puntos) En la siguiente figura se muestra el esquema de un computador **con palabras y direcciones de 32 bits y direccionamiento a nivel de palabra**, que incluye dos buses internos al procesador (datos y direcciones), cuatro registros transparentes (T1, T2, T3 y T4) y un registro acumulador (AC). El incremento o decremento de cualquier registro se realiza atraves de la ALU, cuyas operaciones se muestran en el recuadro superior.



a) (20 %) Determine el tiempo de ciclo de reloj de este computador que dispone de unidad de control cableada. Suponga los siguientes tiempos:
Multiplexor y decodificador:

- **1 ut (unidad de tiempo) Lectura o escritura de registro:**
- **2 ut Lectura o escritura del banco de registros:**
- **3 ut Puerta triestado: despreciable**
- **Operación de la ALU: 20 ut**
- **Tiempo de acceso a memoria: 200 ut**

La duración del ciclo de reloj viene determinada por el camino crítico de esta CPU. El camino crítico es *Reg-ALU-BRegs*, es decir, realizar una operación aritmética con los registros T1 y T2 y almacenar el resultado en el banco de registros:

$$T_{ck} = T_{Reg} + T_{ALU} + T_{Tri} + T_{BR} = 2ut + 20ut + 0 + 3ut = 25ut$$

Como el t_{acc} a memoria es de 200 ut, necesitará 8 ciclos de reloj.

b) (70 %) Desglose, a nivel RT, la instrucción de dos palabras: SWAP .R2, /dir, finalizando con el fetch de la siguiente instrucción. Esta instruccíon realiza el intercambio entre un registro y una posición de memoria, cuya direccion esta contenida en la segunda palabra de la instrucción.

A continuación se realiza el desglose en operaciones elementales de la instrucción propuesta.

La instrucción SWAP intercambia el contenido del registro con el contenido de la dirección de memoria contenida en la segunda palabra de la instrucción.

```
a01: {AR, T2} ← PC              ;La dir está en la 2 palabra de la instrucción por
                                ;lo tanto toca realizar una 2 lectura en memoria

a02: DR ← M(AR), PC ← T2+1      ;
a03: DR ← M(AR)                 ;
a04: AR←DR                      ;Aquí ya busca el contenido de la dirección de
                                ;memoria para meterlo en R2

a05: DR←M(AR),T4←R2             ;Vemos como copia R2 en t4 para luego meter
a06: DR ← M(AR)                 ;t4 en la dirección de memoria
a07: R2←DR                      ;aquí ya tiene el contenido de /dir en DR y lo
                                mete en R2

a08: DR←T4                      ;Ahora la Copia de R2 que estaba en T4 la
                                mete en /dir para ello lo mete en DR con M(AR)
                                ya que AR seguia apuntando a la /dir
a09: DR → M(AR)                 ;

a10: DR → M(AR), {AR, T2} ← PC  ;Aquí termina la instrucción y realiza la fase de
                                ;fetch de la siguiente instrucción
a11: DR ← M(AR), PC ← T2+1      ;
a12: DR ← M(AR)                 ;
a13: IR←DR                      ;Termina la fase de fetch
```

c) (10 %) Calcule el tiempo de ejecución de la instrucción

La instrucción emplea en ejecutar

Teje = 9+4·7 = 37 ciclos -> $37 \cdot 25 = 925ut$

2 (5 puntos) Un computador representa

sigue las convenciones del estándar IEEE 754 en todo excepto en los tamaños, es decir, usa el mismo tipo de representaciones especiales, representación del exponente, bit implíicito, situación de la coma, representaciones normalizadas y no normalizadas, así como bits de guarda y modos de redondeo. En el formato, el bit superior corresponde al signo, los ocho siguientes al exponente y los siete últimos a la mantisa.

a) (20 %) Determine el rango y resolución del formato.

1. Rango y resolución del formato.

Números normalizados.

Exponente: El estándar IEEE754 representa los exponentes en exceso a $2_{n_i1} - 1$. En este caso será exceso a 127. Como se reserva el exponente máximo (−127) para la representación del cero y los números no normalizados, y el exponente mínimo (+128) para la representación del infinito y las indeterminaciones, el rango de exponente para la representación de números queda: [−126, 127].

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

Números normalizados

1,0000000 → 1

Mantisa: ± 1,1111111 → 2 − 2^{-7}

El rango para números normalizados es: ± [$1 \cdot 2^{-126}, (2 - 2^{-7}) \times 2^{127}$]

Números no normalizados

Exponente: Siguiendo el estándar IEEE754, aunque los números no normalizados se representan con el exponente todo a ceros (valor −127), a todos ellos se les asigna como exponente el más pequeño de los normalizados, en este caso −126. De este modo hay continuidad en la representación.

0,0000000 → 0 Mantisas:
± 0,0000001 → 2^{-7}
0,1111111 → 1 − 2^{-7}
El rango para números no normalizados es: $[2^{-7} \cdot 2^{-126}, (1-2^{-7}) \times 2^{-126}] \cup 0$

Así pues, el rango total es:
$\pm [1 \cdot 2^{-126}, (2-2^{-7}) \times 2^{127}] \cup \pm [2^{-7} \cdot 2^{-126}, (1-2^{-7}) \times 2^{-126}] \cup 0$

La resolución depende del exponente y es: $2^{-7} \cdot 2^{E}$

b) (15 %) Determine el valor decimal de los siguientes nu meros que est an representados en el formato:

A = H'BF23 B = H'42D8

Valor decimal de los números.
A = H'BF23 = 1 01111110 0100011 = -1,0100011 · 2^{-1} = -0,10100011 = -0,63671875

- El 2^{-1} Viene de $(2^1+2^2+2^3+2^4+2^5+2^6) - exceso(127) = -1$
- La mantisa son los ultimos 7 digitos
- El símbolo por el primer elemento , 1=- , 0=+

B = H'42D8 = 0 10000101 1011000 = +1,1011000 · 2^6 = +1101100,0 = +108

- $(2^0+2^2+2^7) - exceso(127) = 6$

c) (15 %) Represente en el formato los números C = +7,07 y D = -3× 2^{-126}

3. Paso al formato.

C = +7,07 = +111,00010 = +1,1100010 · 2^2 = 0 10000001 1100010 = H'40E2

- Primero pasar a binario
- Poner un 1 delante de la coma para ver el número normalizado
- 2^2 -> eso significa que nuestro exponente es 127 + 2 = 129
- 129 = 2^7+2^0

D = -3 · 2^{-129} = -11 · 2^{-129} = -1,1000000 · 2^{-128}

Como vemos, al representarlo como número normalizado el exponente se sale de rango. Vemos también que podemos utilizar una representación no normalizada. Basta con forzar el exponente a -126.

Osea lo que ha hecho es $-11 \cdot 2^{-129}$ en vez de dejarlo normalizado 1,1 , lo ha dejado 0,011 para así conseguir el -126.

D = $-0{,}0110000 \cdot 2^{-126}$ = 1 00000000 0110000 = H'8030

d) (30 %) Realice paso a paso la operación A + B - C, dejando el resultado en el formato descrito y determine el valor decimal del mismo. Utilice redondeo al más próximo.

4. Operación A + B − C.

Comenzaremos haciendo A+B para después restarle C

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

A = $-1{,}0100011 \cdot 2^{-1}$ y B = $+1{,}1011000 \cdot 2^{6}$

Se restan los exponentes: $E_A - E_B = -1 - 6 = -7$. Hay que desplazar la mantisa de A siete lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

A = $-1{,}0100011 \cdot 2^{-1}$ **->** 0,000000 1 01 00011 ->
A=$-0{,}0000001\ 01\ 1 \cdot 2^{6}$
Como tienen distinto signo habrá que restar la mantisa de B menos el valor absoluto de la mantisa de A.

$1,\ 1011000\ 00\ 0 \cdot 2^{6}$
$-\ 0{,}0000001\ 01\ 1 \cdot 2^{6}$

$1{,}1010110\ 10\ 1 \cdot 2^{6}$ Normalizado
0, 0000000 10 0 Redondeo (bit de guarda 10 por eso se redondea)

$1,\ 1010111\ 00\ 1 \cdot 2^{6}$

A + B = $+1{,}1010111 \cdot 2^{6}$

Ahora hay que restar C = +1,1100010 · 2^2

Se restan los exponentes: $E_{A+B} - E_C = 6 - 2 = 4$. Hay que desplazar la mantisa de C cuatro lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:
C = +0,0001110 00 1 · 2^6

1, 1010111 00 0 · 2^6
− 0,0001110 00 1· 2^6

1,1001000 11 1· 2^6 Normalizado
0, 0000000 10 0 Redondeo

1, 1001001 01 1 · 2^6

- A + B − C = +1,1001001 · 2^6 = 0 10000101 1001001 = H'42C9
- A+B−C = +1100100,1=100,5

e) (20 %) Determine la resolución con que están representados los números A y C. ¿Qué cambios habría que hacer en el formato para que el número C se pudiera representar con la resolución actual del número A?

5. Resolución de A y C y rediseño de formato. El número A se representa en este formato con exponente −1, es decir, que se representa con una resolución.

- $2^{-7} \cdot 2^{-1} = 2^{-8}$

El número C se representa en este formato con exponente 2, es decir, que se representa con una resolución.

- $2^{-7} \cdot 2^{2} = 2^{-5}$

Para que C se pueda representar con la resolución actual de A se debe cumplir, siendo p el número de bits de la mantisa:

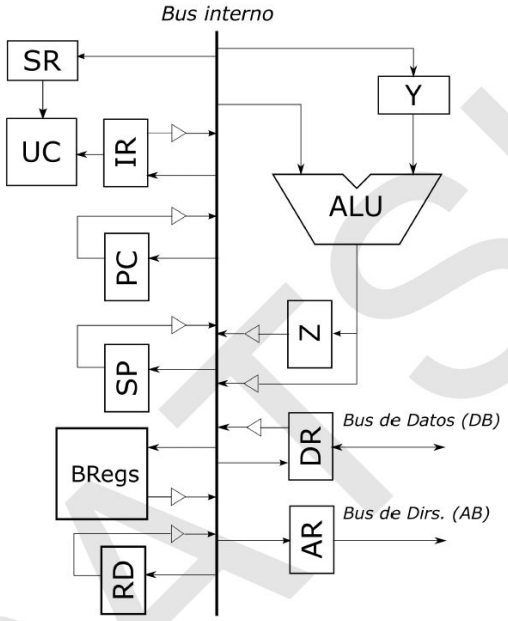
- $2^{-p} \cdot 2^{2} = 2^{-8} \rightarrow 2^{-p+2} = 2^{-8} \rightarrow -p+2=-8 \rightarrow p=10$

O sea, que el único cambio a realizar es usar una mantisa de 10 bits en lugar de los 7 que tiene actualmente.

Enero 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. Su unidad de control es cableada.*

PC: Reg. contador de programa
AR: Reg. de direcciones
IR: Reg. de instrucciones
Y, Z: registros transparentes
BRegs: banco de registros de propósito general, *R0..R7*
RD: es un registro visible al usuario *que se usa sólo para direcciones*
SP: Reg. puntero de pila
DR: Reg. de datos
SR: Reg. de estado



a) Suponiendo que:

- *todas las instrucciones ocupan una sola palabra.*
- *las operaciones elementales duran un ciclo de reloj excepto las que acceden a memoria, que emplean tres ciclos.*
- *el tiempo de ciclo de reloj es 50 ns.*
- *la ALU realiza las operaciones habituales (suma, resta, incremento, decremento, etc.)*
- *la pila crece hacia direcciones decrecientes y SP apunta a la primera dirección libre.*

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

1) el **fetch** *(común a todas las instrucciones, ya que todas ocupan una palabra.)*

2) las siguientes instrucciones (señale con "fetch" la secuencia anterior, supuesta al principio de cada instrucción). Indique claramente con la etiqueta ACTUALIZAR_SR los ciclos en que, en su caso, se deba actualizar el registro de estado, SR.

```
1) LD .R1, [.RD]
```



```
2) ST .R3, [.RD]

3) ADD .R2, .R3, .R4

4) PUSH .R5

5) POP .R5
```

a.2) *De acuerdo al 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 alguna modificación en esta estructura o* datapath *que permita reducir el número de ciclos necesario para la ejecución de las instrucciones propuestas. Haga una estimación de cuánto se reduciría el número medio de ciclos por instrucción.*

- *PC: Reg. contador de programa*
- *AR: Reg. de direcciones*
- *IR: Reg. de instrucciones*
- *Y, Z: registros transparentes*
- *BRegs: banco de registros de propósito general, R0..R7*
- *RD: es un registro visible al usuario que se usa solo para direcciones*
- *DR: Reg. de datos*
- *SP: Reg. puntero de pila*
- *SR: Reg. de estado*

a.1) Una posible descomposición en secuencias de microoperaciones es la siguiente:

Para la fase de fetch es siempre lo mismo cargar el contador de programa en el registro de direcciones, luego sumar uno al PC ("por eso va a Y para llegar a la ALU que es donde se realiza la suma"), Después cuando la instrucción está en AR hacemos la lectura en memoria (para las direcciones en memoria principal siempre a AR), En I+2 vemos que dura dos ciclos y antes ha sumado Y+1 y guardado en PC "ojo ha guardado en Y porque luego tenía que

escribir en PC osea en el mismo registro y hay un solo bus interno". Cuando tengo la dirección cargar los datos del registro de datos por el bus de datos "lógico". Y meto finalmente la dirección en el registro de direcciones. En el enunciado ponen lo que dura el tiempo de lectura a memoria.

Osea la fase de fetch es coger el PC la dirección de la siguiente instrucción y cogerla de memoria para meterla en el registro de instrucciones.

1) fetch:

```
i:      AR <-  PC
        Y  <-  PC
i+1:    M(AR) <- 1er ciclo de M
        PC <- Y+1
i+2:    M(AR) <- 2o ciclo de M
i+3:    DR <- M(AR) 3er ciclo de M
i+4:    IR <-  DR
```

2) instrucciones:

1) LD .R1, [.RD]

Todas parten de la fase de fetch que pone la instrucción en IR, vale nos fijamos en la instrucción LD quiere cargar la dirección en memoria de RD "por eso los corchetes" en R1, por tanto lo primero buscar RD en memoria "en la fase de fetch vimos cómo para conseguir eso había que ir a AR y de ahí por el bus de direcciones a memoria", entonces desde AR hacemos la petición a memoria. Y los datos llegan a DR por el bus de datos. Y con los datos en DR los metemos en R1.

```
fetch
i+5: AR <-  RD
i+6:  <- M(AR); 1er ciclo de M
i+7:  <- M(AR); 2o ciclo de M
i+8: DR <-  M(AR); 3er ciclo de M
i+9: R1 <-  DR
```

Si hubiera sido LD .R1,RD hubiera sido pasar directamente después del fetch R1<-RD

2) ST .R3, [.RD]

Meter R3 en la dirección de memoria de RD, como lo que queremos es acceder a la dirección RD es la que guardamos en AR, luego metemos R3 en DR y metemos los datos de DR en memoria "contienen R3", y luego meter R3 en la dirección que tenía DR antes.

```
fetch
i+5: AR <-  RD
i+6: DR  <- R3
i+7:  <- DR; 1er ciclo de M
i+8:  <- DR; 2o ciclo de M
i+9: M(AR) <-  DR; 3er ciclo de M
```

3) ADD .R2, .R3, .R4

Como queremos suma R3 y R4 y guardarlo en R2 lo que tenemos que hacer es sumar y luego meterlo , por ellos metemos R3 o R4 en Y, luego sumamos por la Alu pero no

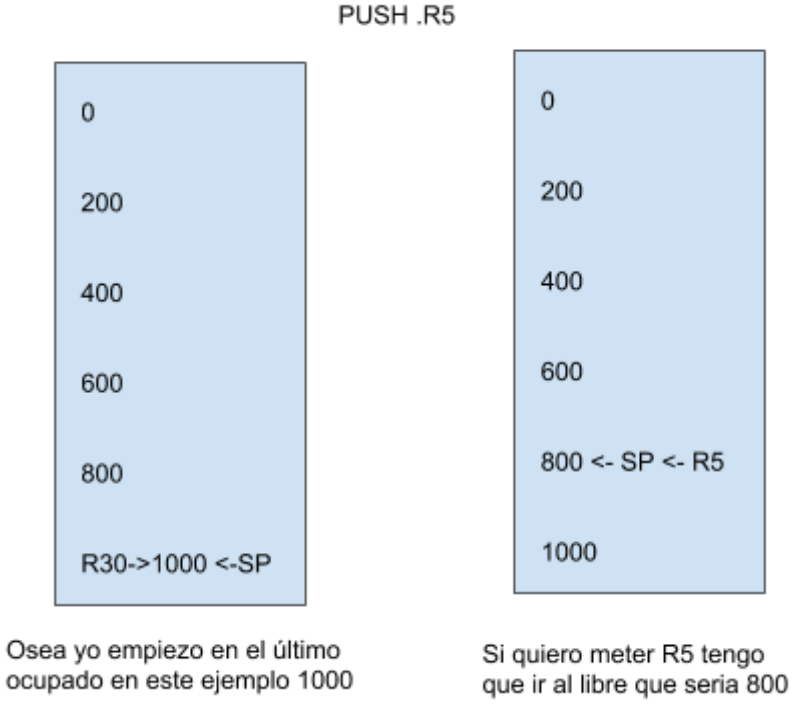
podemos meterlo directamente en R2 porque el bus se ha utilizado para llevar R4 a la Alu. Por eso luego metemos en la siguiente I Z en R2

fetch
i+5: Y <- R3
i+6: Z <- Y + R4; ACTUALIZAR SR
i+7: R2 <- Z

4) **PUSH .R5**

- PUSH y POP
  - **PUSH .R1** — **SP ← SP-4;MEM(SP) ← R1**
  - **POP .R1** — **R1 ← MEM(SP); SP ← SP + 4**

Meter en Pila R5, si nos dicen que SP apunta al último ocupado tendremos que pasar al siguiente como vemos en la imagen de abajo resta para pasar de 1000 el ultimo ocupado a 800.



Cómo vamos a meter en pila lo primero es tener que cuenta SP puntero de pila, lo guardamos en AR y en Y para restarlo, y metemos el Registro en DR. metemos en memoria DR, y restamos uno al puntero de pila, y guardamos DR en la memoria de la dirección.

fetch
i+5: AR <- SP
Y <- SP
i+6: DR <- R5
i+7: <- DR; 1er ciclo de M
SP <- Y - 1
i+8: <- DR; 2o ciclo de M

i+9: M(AR) <- DR; 3er ciclo de M

5) **POP .R5**

En caso del POP buscamos sacar de la pila por tanto sumamos una posición a SP, y metemos la información de esa posición en R5.

fetch
i+5: Y <- SP
i+6: AR <- Y + 1
SP <- Y + 1
i+7: <- M(AR); 1er ciclo de M
i+8: <- M(AR); 2o ciclo de M
i+9: DR <- M(AR); 3er ciclo de M
i+10: R5 <- DR

a.2) De acuerdo al 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.

Mirar los ciclos y el tiempo que necesita por ciclo

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

fetch: 5 ciclos, 250 ns
LD: 10 ciclos, 500 ns
ST: 10 ciclos, 500 ns
SUB: 8 ciclos, 400 ns
PUSH: 10 ciclos, 500 ns
POP: 11 ciclos, 550 ns

a.3)Indique alguna modificación en esta estructura o datapath que permita reducir el número de ciclos necesario para la ejecución de las instrucciones propuestas. Haga una estimación de cuánto se reduciría el número medio de ciclos por instrucción.

Una mejora inmediata y sustancial al datapath original del ejercicio sería la conexión directa del bus de datos al bus interno, lo que permitir´ıa 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 producir´ıa una reducción en el n´umero 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 obtendrá una reducción de al menos un ciclo, 50 ns, en todas ellas gracias al ciclo de menos en el fetch. Además, en el caso de la instrucciones que leen de memoria –LD y POP– se ahorrar´ıa otro ciclo adicional al poderse hacer visible el contenido del DB directamente en el registro destino.

Mantisas Coma fija(comp a 1) y flotante

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

- **Formato 1.-** *Coma fija: 16 bits en complemento a uno con la coma situada en el centro*
- **Formato 2.-** *Coma flotante: 16 bits, con el bit superior para el signo, los ocho siguientes para el exponente (en exceso a 128) y los 7 siguientes para la magnitud de la mantisa, representada en signo-magnitud, con bit implícito y la coma a la izquierda de éste.*

a) Determine el rango y resolución de ambos formatos.

b) Dados los números decimales A = 117,75 y B = -2,453 represéntelos en ambos formatos.

c) Realice paso a paso la operación A+2B en ambos formatos, expresando el resultado en el formato de partida. En el caso de coma flotante utilice un bit de guarda, un bit retenedor y redondeo por forzado a uno.

d) Determine el error producido en las operaciones anteriores.

a)

Rango y resolución

En este caso nos comenta que la coma se encuentra en el centro.

Entonces el primer número como es logico sera todos a 0 y el máximo todos a uno menos el 1, ojo en este caso tenemos 7 a la izquierda de la coma para usar como números , uno para el símbolo y a la derecha 8. El rango por tanto será de 0 a $2^7 - 2^{-8}$. entonces porque luego pone el rango de $-2^7 - 2^{-8}$ porque el primer bit indica el signo de modo que podemos ir de un extremo a otro cambiando el primer bit. La resolución es la diferencia entre los valores de un número representable y el inmediato siguiente(osea en este caso se ve rápido hacíamos $2^7 - 2^{-8}$ pues 2^{-8} sería nuestra diferencia osea imagina de 1,00 a 2,00 hay 0,01 , 0,02… pues en nuestro caso la diferencia serán todos los bits de la derecha que recordamos que los bits de la derecha de ponen en-).

Formato 1 -coma fija, complemento a uno :

$\pm\ 00000000,00000000 \rightarrow 0$

$\pm 01111111,11111111 \rightarrow 2^7 - 2^{-8}$

Rango $= \pm\ [0, 2^7 - 2^{-8}][-(2^7 - 2^{-8}), 2^7 - 2^{-8}]$

Resolución $= 2^{-8}$

Formato 2 -Coma flotante

[1 bit signo | 8 exponente | 7 mantisa]

El exponente va de [-128,127]. Como hay que reservar el -128 para el 0, el rango del exponente queda [-127,127].

Aquí lo primero es dibujar lo que tiene y fijarnos en el exponente nos dice que tiene 8 bits $2^8 = 256$ pero nos dice que tiene exceso 128 osea 256-128 = 128. Ahora hay que tener en cuenta dos cosas que tenemos que reservar el 0 lo reservamos en -128.

$$\text{Mantisa: } \pm \begin{cases} ,10000000 & \rightarrow & 2^{-1} \\ ,11111111 & \rightarrow & 1-2^{-8} \end{cases}$$

$$Rango = \pm [2^{-1} \cdot 2^{-127}, (1-2^{-8}) \cdot 2^{127}] \cup 0$$

$$Resolucion = 2^{-8} \cdot 2^{E}$$

Matisa se trata como el ejercicio anterior el más grande en este caso será el de todos 1 y el más pequeño todos a uno (ojo tenemos 7 bits pero hace $1-2^{-8}$ *porque es mejor eso que sumar todos los bit*),

El rango sigue $[min\ mantisa * 2^{-exponente}, max\ mantisa * 2^{exponente}] \cup 0$

Resolución igual que antes pero en coma flotante se añade el 2^{e}

B)Dados los números decimales A = 117,75 y B = -2,453 represéntelos en ambos formatos.

A = +117,75 = H' 75,= 01110101,1100

0,75 -> 0,1100 -> 12 -> C

Para calcular de un decimal a binario con coma -> 0,75 empiezo multiplicando 0,75*2

1. 1,5 -> 0,5*2 =
2. 1 -> como no hay , todo lo siguiente es 0
3. 0
4. 0

117 → H' 75 → 01110101

De decimal a hexadecimal

Para pasar de base 10 a 16 lo que hacemos es buscar el número multiplicado por 16 más cercano a 117 que en este caso es 7, ese número es 112 y la diferencia es 5, como ese número es menor que 16 es el último que cogemos

117 = X * 16 -> 7

117-112= 5 = X*16 -> X = 0, con lo que terminamos

Cuando ya esta pasado a hexadecimal -> para pasarlo a binario se realiza tal cual 7-> 0111 , 5-> 0101

B = -2,453 = -010,01110011

$-2 \rightarrow 010$

$-\ 0,453 \rightarrow 01110011$ → el metodo mas facil de sacarlo es así

Viene de :

1. 0,906x2=
2. 1,812 -> 2x0,812=
3. 1,624 ->2x0,624=
4. 1,248 ->2x0,248=
5. 0,496 ->2x0,496=

6. 0,992 ->2x0,992=
7. 1,984 ->2x0,984=
8. 1,968 -> ya es el octavo bit

c) Suma A+2B

Vale para hacer 2B desplaza B un lugar a la izquierda (esto se hace en complemento a 1)

$$B = 11111101,10001100 \rightarrow 2B = 11111011,00011001$$

$$\begin{array}{ll} A & 01110101,11000000 \\ +2B & 11111011,00011001 \\ \hline A+B & 1\ \ 01110000,11011001 \\ +\text{carry} & \qquad\qquad\qquad\qquad 1 \\ \hline A+B & 01110000,11011010 \end{array}$$

$$A + 2B = 01110000,11011010 = \text{H'70DA}$$

Ojo los números viene de

- A= 01110101,11000000
- B= -00000010,01110011 = 11111101,10001100 en coma fija vale con dar la vuelta

La suma como podemos ver se realiza sin problemas y cómo al final hay un bit de más se suma como bit de acarreo.

- 1+1 = 0 pero llevando 1 bit
- 0+0 = 0
- 1+0 = 1

Formato 2

Vale en este caso lo primero será representarlos con sus exponentes

$A = +, 11101011 * 2^7$

$B = -, 10011100 * 2^2$

A viene de **01110101,1100 que es 0 10000111 1101011 ->** $+, 11101011 * 2^7$

El primer bit de signo luego el bit de guarda y luego 8 de exponente, con 7 de mantisa el primer bit está activo porque está en exceso 128.

B viene de -010,01110011 que es 1 10000010 0011100 = $-, 10011100 * 2^2$

Para hacer 2B vale con sumar uno su exponente

$B = -, 10011100 * 2^3$

Formato 2.

Para determinar 2B basta con sumarle uno al exponente de B. Así Los números a sumar son:

$A = +,11101011 \cdot 2^7 \qquad 2B = -,10011100 \cdot 2^3$

La diferecia de exponentes: $E_A - E_{2B} = 7 - 3 = 4$ nos dice que hay que desplazar la mantisa de 2B cuatro lugares a la derecha. Teniendo el bit de guarda y el bit retenedor queda:

$2B = -,00001001\ 1\ 1 \cdot 2^7$

Se realiza la suma de mantisas (resta):

M_A	$+,11101011\ 0\ 0$		
$+M_{2B}$	$-,00001001\ 1\ 1$		
$A+2B$	$+,11100001\ 0\ 1 \cdot 2^7$		*Normalizado*
$A+2B$	$+,11100001 \cdot 2^7$		*Redondeo F. a 1*

$A + 2B = +,11100001 \cdot 2^7 = 0\ \ 10000111\ \ 1100001 = H'43E1$

Una vez con los números claros miramos los exponentes como vemos la diferencia es 4 y eso indica el número de puestos que hay que mover la mantisa. La cosa es que al mover eso el bit de guarda y el bit retenedor pueden quedarse activos cosa que en el A no.

- 1-1=0
- 0-1=1 te llevas 1
- 0-0=0
- 1-0=1

d) Determine el error producido en las operaciones anteriores.

A+2B = 117,75-4,906

Formato 1.

El resultado de la suma en coma fija ha sido:

$A + 2B = 01110000,11011010 = 112,8515625$

El error aboluto:

$e_A = ||112,844 - 112,8515625|| = 0,00756$

Formato 2.

El resultado de la suma en coma flotante ha sido:

$A + 2B = +,11100001 \cdot 2^7 = +1110000,1 = 112,5$

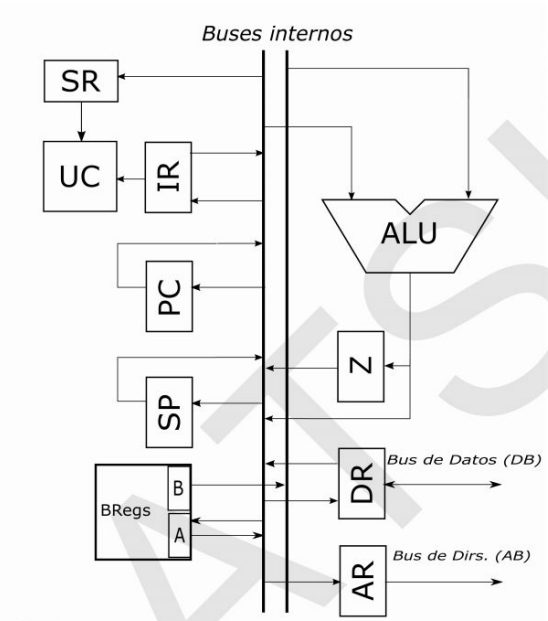
El error aboluto:

$e_A = ||112,844 - 112,5|| = 0,344$

2018 MAYO

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. El banco de registros dispone de dos puertas, A y B, que permiten a la UC seleccionar cualquier pareja de registros en cada ciclo. La memoria es direccionable a palabra, y su tiempo de acceso es de 50 ns. La pila crece en direcciones decrecientes y SP apunta al último dato.

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) Calcule el tiempo de ciclo de reloj para este procesador cuya UC es cableada. Los retardos de algunos dispositivos del computador son:

ALU: 15 ns L/E a registro: 2 ns L/E a banco de registros: 4 ns

b) Desglose en operaciones elementales a nivel RT la instrucción `CALLC #7[.R1], #3[.R3]` incluyendo el fetch de la siguiente instrucción y tratando de solapar operaciones elementales. Esta instrucción de una palabra salta a la subrutina de la primera dirección o a la subrutina de la segunda dirección dependiendo de si se cumple la condición C (carry) o no (NC, not carry), respectivamente.

c) Calcule el tiempo medio de ejecución de la instrucción anterior, suponiendo que la probabilidad de que se cumpla la condición C es del 50 %

d) Explique brevemente la influencia en el tiempo total de ejecución del uso de direccionamiento directo a memoria en ambos campos de dirección de la instrucción anterior: `CALLC /dir1, /dir2`

a) La duración del ciclo de reloj viene determinado por el camino crítico de esta CPU. El camino crítico es BRegs-ALU-BRegs:

$$T_{ck} = T_{BR} + T_{ALU} + T_{BR} = 4\ ns + 15\ ns + 4\ ns = 23\ ns$$

CAMINO CRÍTICO: camino de máximo retardo entre un origen y un destino. Depende de los dispositivos que tengan que atravesar las señales.
En este caso considera el camino más rápido registro + Alu + registro

La operación que piden es CALLC debe salvar la dirección de retorno en pila y dependiendo de si cumple la condición modifica el PC

1. $SP \ -> Z$
2. Z-1 -> SP,AR
3. PC -> DR
4. DR -> M(AR)
5. (SI C) BR(R1) + IR.DESP1
6. (SI NC) BR(R3) + IR.DESP2
7. Z -> PC + AR
8. M(AR) -> DR; PC+1 ->Z
9. Z-> PC
10. DR -> IR

Como el t_{acc} a memoria es de 50 ns, necesitará 3 ciclos de reloj.

b) A continuación se realiza el desglose en operaciones elementales de la instrucción propuesta.

La instrucción CALLC debe salvar la dirección de retorno en pila, y dependiendo de si se cumple la condición C o no, modificar el PC con una u otra dirección.

```
a1:          SP → Z
a2:          Z-1 → SP, AR
a3:          PC → DR
a4:          DR → M(AR) ; dura 3 ciclos
a5: (Si C)   BR(R1) + IR.desp1 → Z
a5: (Si NC)  BR(R3) + IR.desp2 → Z
a6:          Z → PC, AR
a7:          M(AR) → DR; PC+1 → Z (en 1er ciclo);Z → PC (2° ciclo); dura 3 ciclos
a8:          DR→ IR; ir a CO
```

c) La instrucción emplea en ejecutar

$Teje = 8 + 2 \cdot 2 = 12 \ ciclos = 12 \cdot 23 = 276 \ ns$

d) Si la instrucción tuviera los campos de dirección con direccionamiento directo, debería ocupar 3 palabras en total, y habría que leer en memoria esas palabras (al menos la dirección a cargar en el PC), por lo que tardaría como poco tres ciclos más en ejecutar, por lo que sería más lenta.

Otro ejemplo mismo CPU

La cosa de este CPU es la posibilidad de pasar dos cosas por el bus de datos por eso en mucho casos tiene dos acciones por ciclo. Fetch como siempre guardar PC e incrementarlo, llamar de memoria la dirección de la instrucción con M(AR), guardar la instrucción de memoria a DR y de DR a IR.

Fetch

1. AR <- PC
   Z <- PC+1
2. <- M(AR)
   PC <- Z
3. DR <- M(AR)
4. IR <- DR

ST de R3 en la dirección de memoria de R5 con desp 15, Vale la cosa aquí es realizar el desplazamiento para guardar la dirección de R5 con desp 15 en AR, y guardar los datos de R3 en DR, entonces una vez la dirección en AR y los datos en DR se meten los datos en esa dir de memoria.
Ojo pone IR.desp porque no se puede poner 15 directamente, ese 15 viene de la instrucción y por tanto es necesario sacarlo del IR.

ST .R3, #15[.R5]

FETCH

1. Z <- IR.desp + R5
2. AR <- Z
3. DR <- R3
4. <- DR
5. M(AR) <- DR

Restar R4 - R6 y meterlo en R2, como hay doble bus se pueden pasar las dos instrucciones al mismo tiempo

SUB .R2, .R4, .R6

FETCH

1. Z <- R4 - R6; ACTUALIZAR SR
2. R2 <- Z

Ya hemos visto antes Push y recordamos guardar la última dirección **libre (ojo el anterior ejemplo la SP apuntaba a la última usada)** de SP, decrementar SP, meter R7 en el registro de datos para meterlo en memoria (exactamente en la dirección extraída libre de la pila)

PUSH .R7

FETCH

1. AR <- SP
   Z <- SP - 1
2. DR <- R7
3. <- DR
   SP <- Z
4. M(AR) <- DR

RET

FETCH

1. Z <- SP + 1
2. SP <- Z; actualizar SP
3. AR <- Z
4. <- M(AR)
5. DR <- M(AR)
6. PC <- DR

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.*

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
```

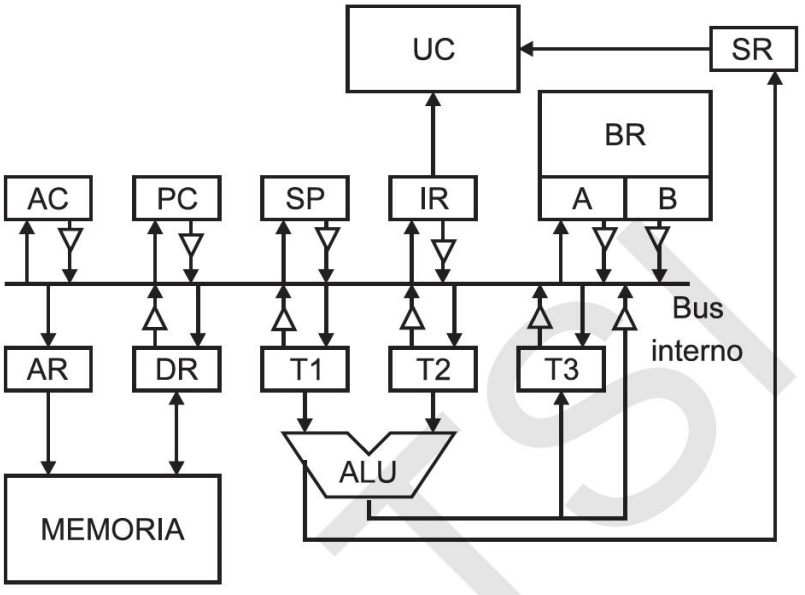
c) Una mejora inmediata, que permitiría ahorra un ciclo en todas lecturas de memoria, sería que hubiese un camino directo desde el bus de datos hasta el interior de la CPU y evitando el registro de datos, DR. Este cambio significaría un ciclo menos en el fetch de todas las instrucciones y otro ciclo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

JULIO 2017 Procesador Tipo 2

1 **(5 puntos)** *Se presenta un esquema muy simplificado de la estructura de un procesador de 64 bits, en donde no aparecen ninguna de las señales de control. La Unidad de control es cableada y se dispone de registro acumulador (AC), tres registros transparentes (T1, T2 y T3) y banco de registros (BR) con dos puertos de salida y uno de entrada. Los accesos a memoria tienen una duración de dos ciclos de reloj y el direccionamiento es a nivel de byte. La pila crece hacia direcciones de memoria decrecientes y el puntero de pila (SP) apunta a la última dirección libre de la cima de la pila. Este procesador trabaja a una frecuencia de 1 GHz.*



a) **(30 %)** *Enumere y explique cuáles son las señales de control necesarias en los siguientes elementos del computador y, para cada una de ellas, indique si deben ser activas por flanco o por nivel:*

- *Banco de registros.*
- *Registros específicos y transparentes.*
- *ALU*
- *Puertas triestado*
- *Memoria.*

b) **(40 %)** *Dadas las instrucciones de una palabra:*

```
CALL [.R2++]
POP [--.R3]
```

exprese a nivel RT (transferencia entre registros) las operaciones elementales que se realizan en cada ciclo de reloj durante la ejecución de cada una de ellas. Incluya la fase de fetch.

c) **(30 %)** *Responda razonadamente a las siguientes preguntas:*

c.1) *¿Cuántos accesos a memoria se realizan al ejecutar cada una de las dos instrucciones propuestas y de qué tipo son estos?*

c.2) *¿Cuál es el tiempo de acceso de la memoria?*

c.3) *¿Cuál es el tiempo de ejecución de cada una de las instrucciones?*

A)

Banco de registros: Dispondrá de una señal de carga de registro, activa por flanco, y de dos grupos de señales, activas por nivel, que seleccionarán cada uno de los puertos A y B.

Registros específicos y transparentes: Cada registro precisa una señal de carga, salvo el registro de datos de memoria que dispondrá de dos. Todas ellas serán activas por flanco.

ALU: Tendrá como entrada un grupo de señales que controlarán la operación a realizar. Activas por nivel.

Puertas triestado: Cada triestado tiene una señal de activación por nivel.

Memoria: La memoria dispondrá, al menos, de una señal de selección y de otra que indique lectura o escritura. Activas por nivel.

flecha normal activa por flanco , flecha blanca activa por nivel.
Como siempre guardar pc en AR, y sumar uno al PC (64 bits las dir por eso suma 8), ojo si es unidad de control programada necesita decodificación en el fetch
FETCH :

1. AR,T1 <- PC
2. Acceso a memoria , $PC <- T1 + 8;$ leer instrucción
3. DR <- M(AR)
4. IR <- DR
5. DECODIFICACIÓN

Recordamos Call lo que hace es llamar a R2 en este caso con postdecremento de R2
Es fácil ver que lo que está haciendo es meter r2 en pc
CALL [.R2 ++]

1. FETCH
2. T1, AR <- SP
3. DR <- PC
4. ACCESO A MEMORIA, SP <- T1 - 8

5. M(AR) <- DR, T1 , PC <- R2
6. R2 <- T1 + 8

POP [--,R3]

1. FETCH
2. T1 <- SP
3. SP, AR <- T1+8
4. Acceso a memoria, T1 <- R3
5. DR <- M(AR), R3 <- T1-8
6. AR <- R3
7. Acceso a memoria
8. M(AR) <- DR

Otro ejemplo (3 ciclos de lectura de reloj)(microprogramada)

MOVE_Z [.R5++], [++.R7]

1. SI NZ MICROSALTO A FETCH
2. AR,T1 <- R5
3. Acceso a memoria , R5 <- T1+8
4. Acceso a memoria, T2 <- R7
5. DR <- M(AR)
6. AR , R7 <- T2+8
7. ACC MEM
8. ACC MEM
9. M(AR) <- DR, microsalto a fetch

c) En la instrucción CALL se realizan dos accesos a memoria: una lectura y una escritura. La lectura corresponde al fetch y la escritura a la salvaguarda de la dirección de retorno. Para la instrucción POP hay tres accesos a memoria: dos lecturas y una escritura. Además de la lectura debida al fetch, hay otra lectura del dato almacenado en la pila que después se escribe en memoria, en la dirección que indica la instrucción.

La frecuencia de 1 GHz corresponde a un periodo de reloj de 1ñs. Como el enunciado indica que los accesos a memoria son de dos ciclos de reloj, el tiempo de acceso de la memoria es de 2ñs.

El tiempo de ejecución de una instrucción es el número de ciclos que utiliza, multiplicado por el tiempo de cada ciclo. Por lo tanto:

tCALL = 10 ciclos . 1ñs/ciclo = 10ñs

tPOP = 12 ciclos . 1ñs/ciclo = 12ñs

ARITMETICA IEEE754

2 **(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 7 siguientes el exponente y los 8 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 la resolución del formato de representación.*

b) *Represente en este formato el número decimal A = +5,05*

c) *Dadas la cadenas de bits B = H'BE74, y C = H'8014 que representan números en el formato citado, determine su valor decimal.*

d) *Realice paso a paso la operación 5A + B, usando redondeo al más próximo y dejando el resultado en el formato de almacenamiento en memoria.*

e) *Determine el error absoluto cometido en la operación.*

a) <u>Rango y resolución</u>

<u>Normalizados</u>

IEEE754 representa el exponente en exceso a $2^{n-1}-1$. En este caso sería exceso 63.

Como se reserva exponente mínimo (-63) para la representación del cero y los números no normalizados, y el exponente máximo (+64) para la representación del infinito y las indeterminaciones, el rango de exponente sera [-62,63].

En el formato tienen signo-magnitud, bit implícito y la coma a la derecha del bit implícito:

Mantisa :

$\{1,00000000 \rightarrow 1$

$\{1,11111111 \rightarrow 2-2^{-8}$

El rango para números es $\pm[1*2^{-62},(2-2^{-8})*2^{63}]$

<u>No normalizados</u>

Estos se representan con exponente todo a 0, todos ellos tienen como exponente el más pequeño de los normalizados en este caso -62.

Mantisa:

$\{0,00000000 \rightarrow 0$

$\{0,00000001 \rightarrow 2^{-8}$

$\{0,11111111 \rightarrow 1-2^{-8}$

El rango para números no normalizados es $\pm[2^{-8}*2^{-62},(1-2^{-8})*2^{-62}]\cup 0$

Resolución : $2^{-8}*2^{e}$

<u>b)Representación</u>

A= $+5,05 = +101,00001100 = 1,01000011*2^{2} = 0\ 1000001\ 01000011 = H'4143$

<u>c)Valor decimal</u>

B = $H'BE74 = 1\ 0111110\ 01110100 = -1,01110100*2^{-1} = -0,10111010 = -0,7265625$

C = $H'8014 = 1\ 0000000\ 00010100$ es no normalizado sería así :
$-0,00010100 * 2^{-62} = -101,00 * 2^{-68} = -5 * 2^{-68}$

d) Suma 5A + B
$1,01000011 * 2^{2}$
$-1,01110100 * 2^{-1}$

5A = 4A + A
4A es incrementar dos veces el exponente = $1,01000011 * 2^{4}$
4A + A
Primero mirar exponentes
4-(2)= 2 ahora desplazar la mantisa del menor dos posiciones y poner 2 bit guarda y 1 de redondeo

$1,01000011\ 000$
$+0,01010000\ 110$

$1,10010011\ 110 * 2^{4}$
$+0,00000000\ 100$

$5A = 1,10010100\ 010 * 2^{4}$

Ahora sumar 5A + B
Primero exponentes 4 -(-1)=5
B = $-1,01110100 * 2^{-1}$ -> $-0,00001011\ 101 * 2^{4}$

$1,10010100\ 000$
$-0,00001011\ 101$

$1,10001000\ 011$
$-0,00000000\ 100$

$1,10001000\ 111$
5A + B = $1,10001000 * 2^{4} = 0\ 1000011\ 10001000 = H'4388$
E) Error cometido en la operación

5A + B= 25,25 - 0,7265625 = 24,5234375
5A + B= $1,10001000 * 2^{4}$ = 11000,1 = 24,5
Para pasar simplemente a desplazado la coma 4 posiciones y luego ha pasado de binario a decimal $2^{3} + 2^{4} = 24 + 2^{-1} = 24,5$
Ea = 0,0234

Aritmética IEEE754 ejemplo 2

2 **(5 puntos)** Un computador representa números de coma flotante usando un formato de 16 bits que sigue las convenciones del 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 modos de redondeo. En el formato, el bit superior corresponde al signo, los nueve siguientes al exponente y los seis últimos a la mantisa.

a) Determine el rango y resolución del formato.

b) Represente en el formato los números A = -0,64 y B = +108

c) Determine el valor decimal de los siguientes números que están representados en el formato: C = H'4131; D = H'8018

d) Realice paso a paso la operación A + B + C, dejando el resultado en el formato descrito y determine el valor decimal del mismo. Utilice redondeo al más próximo.

e) Rediseñe el formato de representación (manteniendo el número total de bits) para que el número A se pueda representar con una resolución menor que 10^{-3}

SOLUCIÓN

1. **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:

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

   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), a todos ellos se les asigna como exponente el más pequeño de los normalizados, en este caso -254. De este modo hay continuidad en la representación.

Mantisas: $\pm \begin{cases} \texttt{0,000000} & \rightarrow & 0 \\ \texttt{0,000001} & \rightarrow & 2^{-6} \\ \texttt{0,111111} & \rightarrow & 1-2^{-6} \end{cases}$

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

Así pues, el rango total es:

$\pm\left[1\cdot 2^{-254}, (2-2^{-6})\cdot 2^{255}\right] \cup \pm\left[2^{-6}\cdot 2^{-254}, (1-2^{-6})\cdot 2^{-254}\right] \cup 0$

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

2. **Paso al formato.**

   $A = \texttt{-0,64} = \texttt{-,10100011} = \texttt{-1,010001}\cdot 2^{-1} = \texttt{1 011111110 010001} = \texttt{H'BF91}$

   $B = \texttt{+108} = \texttt{+01101100} = \texttt{+1,101100}\cdot 2^{6} = \texttt{0 100000101 101100} = \texttt{H'416C}$

3. **Valor decimal de los números.**

   $C = \texttt{H'4131} = \texttt{0 100000100 110001} = \texttt{+1,110001}\cdot 2^{5} = \texttt{+111000,1} = \texttt{+56,5}$

   $D = \texttt{H'8018} = \texttt{1 000000000 011000}$

   Como se puede ver, el exponente del número D es cero, con lo cual representa un número no normalizado, al que se le asigna el exponente mínimo (-254).

   $D = \texttt{-0,011000}\cdot 2^{-254} = \texttt{-11,0}\cdot 2^{-257} = \texttt{-3}\cdot 2^{-257} = \texttt{-1,30}\cdot 10^{-77}$

4. **Suma $A+B+C$.**

   Comenzaremos haciendo A+B para después sumarle C

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

   $A = \texttt{-1,010001}\cdot 2^{-1}$ *y* $B = \texttt{+1,101100}\cdot 2^{6}$

   Se restan los exponentes: $E_A - E_B = -1-6 = -7$. Hay que desplazar la mantisa de A siete lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

   $A = \texttt{-0,000000 10 1}\cdot 2^{6}$

   Como tienen distinto signo habrá que restar la mantisa de B menos el valor absoluto de la mantisa de A.

M_B	$1,101100\ 00\ 0\cdot 2^{6}$	
M_A	$-\ 0,000000\ 10\ 1\cdot 2^{6}$	
	$1,101011\ 01\ 1\cdot 2^{6}$	*Normalizado*
Redondeo	$0,000000\ 10\ 0$	
	$1,101011\ 11\ 1\cdot 2^{6}$	

   $A+B = \texttt{+1,101011}\cdot 2^{6}$

   Ahora hay que sumar $C = \texttt{+1,110001}\cdot 2^{5}$

   Se restan los exponentes: $E_{A+B} - E_C = 6-5 = 1$. Hay que desplazar la mantisa de C un lugar a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

   $C = \texttt{+0,111000 10 0}\cdot 2^{6}$

M_{A+B}	$1,101011\ 00\ 0\cdot 2^{6}$
M_C	$+\ 0,111000\ 10\ 0\cdot 2^{6}$
	$10,100011\ 10\ 0\cdot 2^{6}$
Normalizacion	$1,010001\ 11\ 0\cdot 2^{7}$
Redondeo	$0,000000\ 10\ 0$
	$1,010010\ 01\ 0\cdot 2^{7}$

   $A+B+C = \texttt{+1,010010}\cdot 2^{7} = \texttt{0 100000110 010010} = \texttt{H'4192}$

   $A+B+C = \texttt{+10100100} = 164$

5. **Rediseño de formato.**

El número A se representa en este formato con exponente -1, es decir, que se representa con una resolución $2^{-6} \cdot 2^{-1}$. En general, para p bits de mantisa su resolución será $2^{-p} \cdot 2^{-1}$.

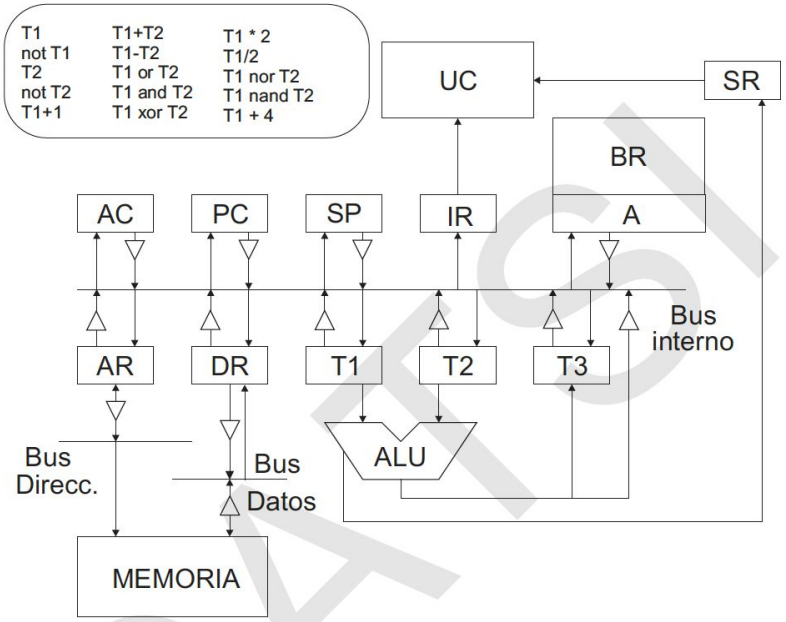
Así pues se debe cumplir:

$2^{-p} \cdot 2^{-1} < 10^{-3} \Rightarrow 2^{-(p+1)} < 10^{-3} \Rightarrow p+1 > 3/log2 \Rightarrow p+1 > 9,96$

Por tanto, para cumplir la restricción el número de bits de la mantisa debe ser 9. Nos quedaría un formato con un bit para el signo, seis para el exponente y nueve para la mantisa.

JULIO 2016 procesador con operaciones est

1 ***(5 puntos)*** *En la figura se muestra el esquema de la estructura de un computador con palabras y direcciones de 32 bits. El procesador dispone de Unidad de control microprogramada (UC), registro acumulador (AC), tres registros transparentes (T1, T2 y T3) y Banco de registros (BR) con 12 registros generales y un único puerto de entrada y de salida (A). Los accesos a memoria tienen una duración de dos ciclos de reloj y el direccionamiento es a nivel de byte. La pila crece hacia direcciones de memoria crecientes y el puntero de pila apunta a la última dirección ocupada de la cima de la pila. Los incrementos o decrementos de registros se realizan en la ALU; en la parte superior de la figura, se indican sus operaciones aritméticas y lógicas.*



a) **(15 %)** *Realice a nivel RT (transferencia entre registros) el microprograma de la microsubrutina de fetch ejecutada en este computador.*

b) **(60 %)** *La instrucción de dos palabras* `CALL_NZ [.R4], [.R6], /4000` *realiza un salto a subrutina si son distintos los operandos especificados por los dos primeros direccionamientos. La dirección de comienzo de la subrutina corresponde al tercer direccionamiento y está contenida en la segunda palabra de la instrucción. Exprese a nivel RT las microinstrucciones que se realizan en cada ciclo de reloj durante la fase de ejecución.*

c) **(25 %)** *Calcule el tiempo medio de ejecución de la instrucción anterior teniendo en cuenta que la probabilidad de que la condición se verifique es del 70 %. Considere tiempos despreciables de carga o lectura de registros y los siguientes retardos en los elementos del procesador:*

Secuenciador del microprograma: 5 ut (unidades de tiempo)
Memoria de control: 40 ut
Operación más lenta de la ALU: 20 ut
Carga o lectura del banco de registros: 2 ut
Triestado: 1 ut

SOLUCIÓN

a) A continuación se muestra el microprograma de la microsubrutina de fetch, expresada a nivel RT.

```
f1:AR ← PC
f2:Acceso a memoria, T1 ← PC
f3:DR ← M(AR), PC ← T1+4


f4:IR ← DR, microsalto a C.O.
```

b) Seguidamente se relacionan a nivel RT las microinstrucciones ejecutadas, en cada ciclo de reloj, en el microprograma que corresponde a la fase de ejecución de la instrucción `CALL_NZ [.R4], [.R6], /4000`

```
m1:AR ← R4
m2:Acceso a memoria
m3:DR ← M(AR); primer operando
m4:AR ← R6
m5:Acceso a memoria, T1 ← DR
m6:DR ← M(AR); segundo operando
m7:T2 ← DR
m8:T3 ← T1-T2, actualizar SR
m9:Si Z microsalto a m20
m10:T1, AR ← PC
m11:Acceso a memoria
m12:DR ← M(AR); dirección subrutina
m13:T3 ← DR
m14:DR ← T1+4; dirección de retorno
m15:T1 ← SP
m16:AR, SP ← T1+4
m17:Acceso a memoria; guardar dirección de retorno
m18:M(AR) ← DR
m19:PC ← T3, microsalto a fetch
m20:T1 ← PC
m21PC ← T1+4, microsalto a fetch
```

c) Para el cálculo del tiempo de ejecución se debe calcular, en primer lugar, el número medio de ciclos considerando que la probabilidad de salto a subrutina es del 70 %.

$$N^o\ medio\ de\ ciclos\ =\ ciclos_{fetch} + ciclos_Z + ciclos_{NZ}\ =\ (4 + 0,3 \cdot 11 + 0,7 \cdot 19)\ ciclos\ =\ 20,6\ ciclos$$

Teniendo en cuenta que el tiempo de ciclo es la suma del tiempo de la operación elemental de mayor duración (tiempo del camino crítico) y del retardo debido a la unidad de control microprogramada:

$$t_{C_{critico}}\ =\ t_{ALU} + t_{triestado} + t_{BR}\ =\ (20 + 1 + 2)\ ut\ =\ 23\ ut$$
$$t_{U_{control}}\ =\ t_{Secuenciador} + t_{M_{control}}\ =\ (5 + 40)\ ut\ =\ 45\ ut$$
$$t_{ciclo}\ =\ (23 + 45)\ ut\ =\ 68\ ut$$

Obsérvese que no se han considerado los retardos debidos a carga o lectura de registros, ya que estos son despreciables.

Finalmente, el tiempo medio de ejecución de la instrucción será:

$$t_{ejec}\ =\ 20,6\ ciclos\ .\ 68\ ut/ciclo\ =\ 1400,8\ ut$$

2 ***(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 la resolución del formato de representación.*

b) *Represente en este formato los números decimales:* $A = 25{,}25$; $B = 8{,}5 \cdot 2^{-260}$

c) *Dada la cadena de bits* $C = H'BF9D$*, que representa un número en el formato citado, determine su valor decimal.*

d) *Realice paso a paso la operación* $A + C$*, usando redondeo al más próximo y dejando el resultado en el formato de almacenamiento en memoria.*

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 de los números.

$A = \texttt{+25,25} = \texttt{+11001,01} = \texttt{+1,100101}\cdot 2^4 = \texttt{0 100000011 100101} = \texttt{H'40E5}$

$B = \texttt{-8,5}\cdot 2^{-260} = \texttt{+1000,1}\cdot 2^{-260}$

El número no se puede representar como normalizado porque el exponente se saldría de rango. Se puede representar como no normalizado. Buscamos el exponente -254.

$B = \texttt{+0,0010001}\cdot 2^{-254} = \texttt{0 00000000 001000} = \texttt{H'0008}$

c) Valor decimal.

$C = \texttt{HBF9D} = \texttt{1 011111110 011101} = \texttt{-1,011101}\cdot 2^{-1} = \texttt{-0,1011101} = \texttt{-0,7265625}$

d) Suma $A + C$.

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

$A = \texttt{+1,100101}\cdot 2^4 \;\; y \;\; C = \texttt{-1,011101}\cdot 2^{-1}$

Se restan los exponentes: $E_A - E_C = 4-(-1) = 5$. 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:

$C = \texttt{-0,000010 111}\cdot 2^4$

$$\begin{array}{lll} M_A & 1,100101\ 000 & \\ M_C & -\ 0,000010\ 111 & \\ \hline & 1,100010\ 001\cdot 2^4 & Normalizado \\ Redondeo & 0,000000\ 100 & \\ \hline & 1,100010\ 101\cdot 2^4 & \end{array}$$

$A + C = \texttt{+1,100010}\cdot 2^4 = \texttt{0 100000011 1000010} = \texttt{H'40E2}$

e) Valor decimal del resultado.

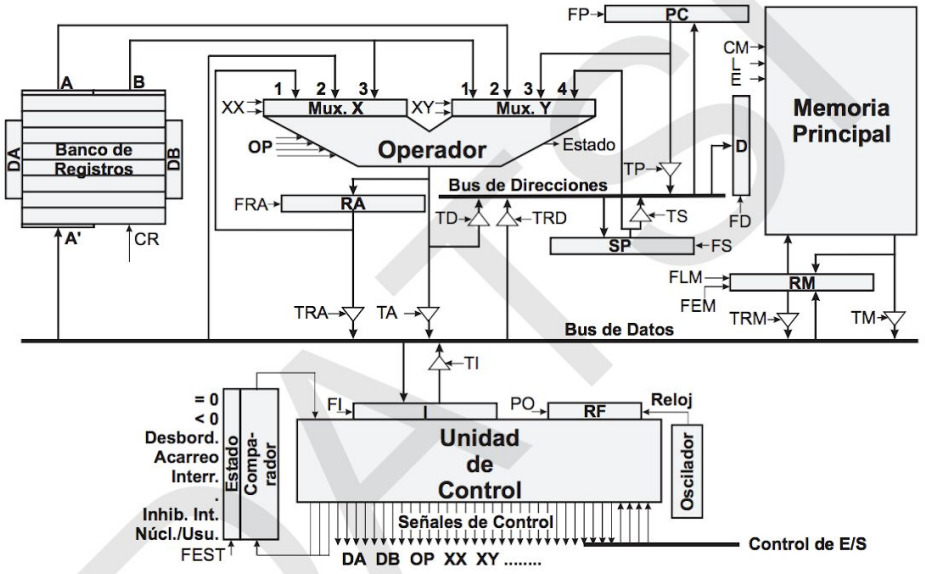
$$A + C = \texttt{+1,100010}\cdot 2^4 = \texttt{+11000,10} = \texttt{+24,5}$$

2016 otro ejemplo

1 ***(5 puntos)*** *En la figura se muestra la estructura de un computador elemental con palabras y direcciones de 32 bits y memoria direccionable a nivel de byte, cuya unidad de control es cableada. D es el registro de direcciones, RM el registro de datos, I el registro de instrucción y RA el registro acumulador. Los tiempos de operación de sus elementos son los siguientes:*

Acceso a memoria: 50 ns	*Lectura o escritura de registros: 0,5 ns*
Operador de la ALU: 5 ns	*Lectura o escritura del banco de registros: 1 ns*
Multiplexores: 2 ns	*Puertas triestado: 0,3 ns*

El operador de la ALU incluye entre sus operaciones aritméticas el incremento en 4 de la entrada seleccionada en el multiplexor Y.



La instrucción de dos palabras `ST /200, #8[.R4 ++]`*, en la que la dirección absoluta de memoria está contenida en la segunda palabra, forma parte del juego de instrucciones.*

a) *Determine el periodo de reloj para este computador. ¿Cuántos ciclos de reloj ocupará cada acceso a memoria?*

b) *Represente en un cronograma las señales de control activadas,* ***en cada ciclo de reloj****, durante la fase de fetch. Para facilitar la representación, considere que los accesos a memoria duran dos ciclos de reloj.*

c) *Especifique mediante notación RT (transferencia entre registros) las operaciones elementales que se producen* ***en cada ciclo de reloj*** *durante la fase de ejecución de la instrucción indicada. Como en el apartado anterior, suponga que los accesos a memoria tienen una duración de dos ciclos de reloj.*

d) *Obtenga el tiempo total de ejecución de la instrucción anterior, incluyendo un ciclo de decodificación.*

SOLUCIÓN

a) El camino crítico o camino de mayor retardo, definido a partir de la operación elemental de mayor duración, determina el periodo de reloj. En este computador es posible realizar, en un mismo ciclo, la lectura de dos registros del banco de registros, su operación en la ALU y la carga del resultado en uno de los registros generales del banco de registros. Por tanto:

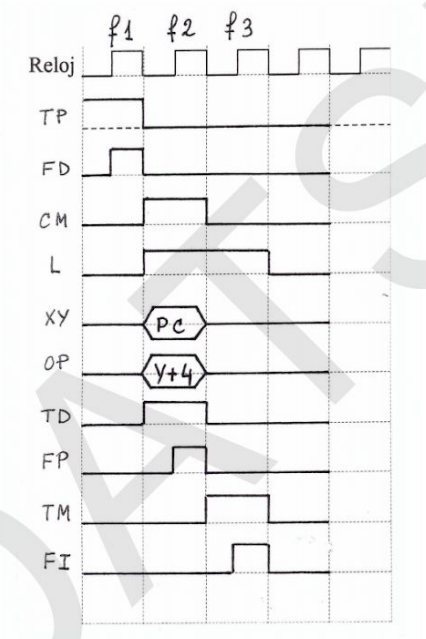
$t_{ck} = t_{lectura_{BR}} + t_{Mux} + t_{ALU} + t_{triestado} + t_{escritura_{BR}} = (1 + 0,3 + 5 + 0,2 + 1)\ ns = 7,5\ ns$

Puesto que el tiempo de acceso a memoria es de 50 ns, cada acceso a memoria es de 7 ciclos de reloj.

b) Considerando, por facilidad en el diseño del cronograma, que los accesos a memoria son de dos ciclos de reloj, expresamos las operaciones elementales correspondientes al fetch:

```
f1:  D<- PC
f2:  Acceso a memoria (lectura)
     PC<- PC + 4; el PC apunta a la segunda palabra
f3:  I<- M(D)
```

Para este fetch, el cronograma, con las correspondientes señales de control activadas en cada ciclo de reloj se muestra a continuación.



c) La instrucción propuesta realiza tres accesos a memoria que consideraremos también de dos ciclos: lectura de la segunda palabra, lectura del dato y escritura del dato. Se muestran, a nivel RT, las operaciones elementales de la fase de ejecución en cada ciclo de reloj:

```
e1: D<- PC; lectura de la segunda palabra
e2: Acceso a memoria (lectura)
    PC<- PC + 4; el PC apunta a la siguiente instrucción
e3: D<- M(D)
e4: Acceso a memoria (lectura)
e5: RM<- M(D)
e6: D<- R4 + I(desp)
e7: Acceso a memoria (escritura)
    R4<- R4 + 4; postincremento del registro base
e8: M(D)<- RM
```

d) El tiempo que tarda en ejecutarse la instrucción incluye los ciclos debidos al fetch, a la decodificación y a la fase de ejecución, en total 12 ciclos:

$t_{ejec} = t_{ciclo} * n^{o}\ de\ ciclos = 7,5\ ns/ciclo * 12\ ciclos = 90\ ns$

2 ***(5 puntos)*** *Un computador representa números de coma flotante usando un formato de 16 bits que sigue las convenciones del 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. En el formato, el bit superior corresponde al signo, los seis siguientes al exponente y los 9 últimos a la mantisa.*

a) *Determine el rango y la resolución del formato.*

b) *Represente en el formato los siguientes valores:*

$$A = +21{,}8 \quad ; \quad B = +\infty$$

c) *Determine el valor decimal de los siguientes números que están representados en el formato:*

$$C = H'B908 \quad ; \quad D = H'8068$$

d) *Realice paso a paso la suma A+C dejando el resultado en el formato de partida. Utilice redondeo al más próximo.*

e) *Determine el error absoluto que se ha producido en la operación.*

SOLUCIÓN

1. **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 31. Como se reserva el exponente mínimo (-31) para la representación del cero y los números no normalizados, y el exponente máximo ($+32$) para la representación del infinito y las indeterminaciones, el rango de exponente para la representación de números queda: $[-30, 31]$.

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

   $$\text{Mantisa: } \pm \begin{cases} \texttt{1,000000000} & \rightarrow \quad 1 \\ \texttt{1,111111111} & \rightarrow \quad 2-2^{-9} \end{cases}$$

   El rango para números normalizados es: $\pm\left[1 \cdot 2^{-30}, (2-2^{-9}) \cdot 2^{31}\right]$

   Números no normalizados

   Exponente: Siguiendo el estándar IEEE754, aunque los números no normalizados se representan con el exponente todo a ceros (valor -31), a todos ellos se les asigna como exponente el más pequeño de los normalizados, en este caso -30. De este modo hay continuidad en la representación.

   $$\text{Mantisas: } \pm \begin{cases} \texttt{0,000000000} & \rightarrow \quad 0 \\ \texttt{0,000000000} & \rightarrow \quad 2^{-9} \\ \texttt{0,111111111} & \rightarrow \quad 1-2^{-9} \end{cases}$$

   El rango para números no normalizados es: $\pm\left[2^{-9} \cdot 2^{-30}, (1-2^{-9}) \cdot 2^{-30}\right] \cup 0$

   Así pues, el rango total es:

   $\pm\left[1 \cdot 2^{-30}, (2-2^{-9}) \cdot 2^{31}\right] \cup \pm\left[2^{-9} \cdot 2^{-30}, (1-2^{-9}) \cdot 2^{-30}\right] \cup 0$

   La resolución depende del exponente y es: $2^{-9} \cdot 2^{E}$

2. **Paso al formato.**

   $A = \texttt{+21,8} = \texttt{+10101,11001100} = \texttt{+1,010111001} \cdot 2^4 = \texttt{0 100011 010111001} = \texttt{H'46B9}$

   $B = +\infty = \texttt{0 111111 000000000} = H'7E00$

3. **Valor decimal de los números.**

   $C = \texttt{H'B908} = \texttt{1 011100 100001000} = \texttt{-1,100001000} \cdot 2^{-3} = \texttt{-0,001100001000} = \texttt{-0,189453125}$

   $D = \texttt{H'8068} = \texttt{1 000000 001101000}$

   Como se puede ver, el exponente del número D es cero, con lo cual representa un número no normalizado, al que se le asigna el exponente mínimo (-30).

   $D = \texttt{-0,001101000} \cdot 2^{-30} = \texttt{-1101} \cdot 2^{-36} = \texttt{-13} \cdot 2^{-36}$

4. **Suma $A + C$.**

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

   $A = \texttt{+1,01011101} \cdot 2^4 \;\; y \;\; C = \texttt{-1,100001000} \cdot 2^{-3}$

   Se restan los exponentes: $E_A - E_C = 4 - (-3) = 7$. Hay que desplazar la mantisa de C siete lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

   $C = \texttt{-0,000000110 00 1} \cdot 2^4$

M_A	$1,010111001\ 00\ 0 \cdot 2^4$	
M_C	$-\ 0,000000110\ 00\ 1 \cdot 2^4$	
	$1,010110010\ 11\ 1 \cdot 2^4$	*Normalizado*
Redondeo	$0,000000000\ 10\ 0$	
	$1,010110011\ 01\ 1 \cdot 2^4$	

   Representación:

   $A + C = \texttt{+1,010110011} \cdot 2^4 = \texttt{0 100011 010110011} = \texttt{H'46B3}$

5. **Error absoluto.**

   $A + C = \texttt{21,8} - \texttt{0,189453125} = \texttt{21,61054688}$

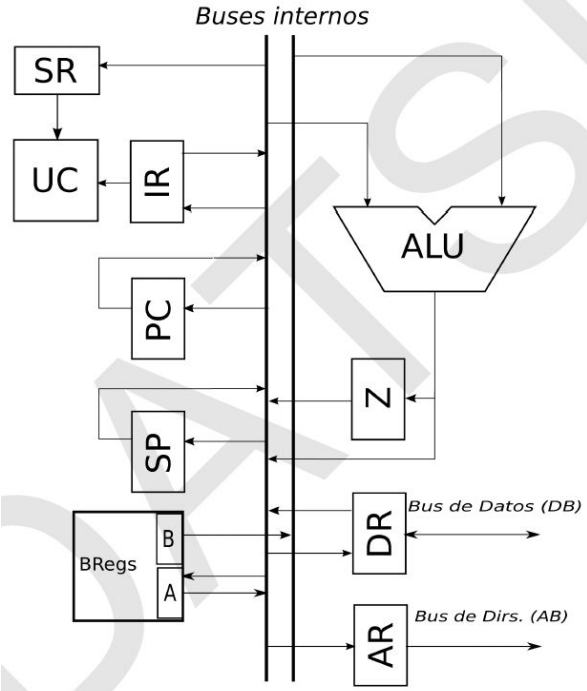
   $A + C = \texttt{+1,010110011} \cdot 2^4 = \texttt{10101,10011} = \texttt{21,59375}$

   $Error = \|\texttt{21,61054688} - \texttt{21,59375}\| = \texttt{0,01679688}$

Julio 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 instrucci ́on
- Z: registro transparente
- BRegs: banco de registros de prop ́osito 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 direcci on 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.)

I. AR-> PC
II. Z <- PC+1
III. <- M(AR) , PC ->Z
IV. DR <- M(AR)
V. IR <- DR

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 - Cargamos el contenido de R3 en memoria en R1 con desplazamiento 13

1) LD .R1, #13[.R3]

IV. Z<- IR.desp + R3 obtenemos R3 + 13
V. AR <- Z

VI. <- M(AR) 1 ciclo lectura
VII. DR <- M(AR)
VIII. R1 <- DR

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

Fetch guardamos R3 en R5 con desplazamiento 12

IV. Z <- IR.desp + R5
V. AR <- Z
VI. DR <- R3
VII. <- DR
VIII. M(AR) <- DR

3) ADD .R2, .R4, .R6

IV. Z <- R4 + R6
V. R2 <- Z

4) POP .R7

IV. Z <- SP+1 nuevo valor SP
V. SP <- Z, actualiza SP AR <- Z
VI. <- M(AR)
VII. DR <- M(AR)
VIII. R7 <- DR

5) RET

I. Z <- SP +1 Nuevo valor SP
II. SP <- Z actualiza SP AR<- Z
III. <- M(AR)
IV. DR <- M(AR)
V. PC <- DR

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

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) Indique si encuentra alguna posible modificación esta estructura o datapath que

permitiese reducir el número de ciclos necesarios para la ejecución de las instrucciones propuestas.

Si se analiza la secuencia de microoperaciones obtenida para cada instrucción, se observa que ser obtendrá 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.

Ejercicio 2

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.

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:

Mantisa positiva : $,1000\ 0000\ 000 \rightarrow 2^{-1}$
$,1111\ 1111\ 111 \rightarrow 1-2^{-11}$

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

b) A = H03CB5 es una representación en este formato. Determine su valor decimal.
A = H03CB5 = 0011 1100 1011 0101 = 0 01111 0010110101= $+,10010110101\cdot 2^{-1}$
A= +,010010110101 = +,0294189453

c) Represente en el formato el número decimal B = −50, 0125.
B = −50, 0125= -110010,00000011=-,110010 00000 $\cdot 2^{6}$
Quedando en formato B= 1 10110 10010 00000

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.

Suma A + B. Los números a sumar son:
A = +,10010110101 $\cdot 2^{-1}$ y B = -,11001000000 $\cdot 2^{6}$

Determinamos la diferencia de exponentes: EA − EB = −1 − 6 = −7. Hay que desplazar la mantisa de A siete lugares a la derecha.

Suma de las mantisas (resta):

$$
\begin{array}{lll}
B & -,11001000000\ \ 000 \cdot 2^6 & \\
A & +,00000001001\ \ 011 \cdot 2^6 & \\
\hline
A+B & -,11000110110\ \ 101 \cdot 2^6 & \text{Normalizado} \\
 & +,00000000000\ \ 100 & \text{Redondeo} \\
\hline
 & -,11000110111\ \ 001 & \cdot 2^6
\end{array}
$$

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

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

e) Rediseñe el formato de representación sin variar el número total de bits para que se pueda representar el número 1010.

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 Emax al mı́ aximo exponente, tendremos que dicho nuḿ ero ser ıa $(1 - 2^{-P}) \cdot 2^{Emax}$. Así,sedebecumplir:$(1-2^{-P}) \cdot 2^{Emax} > 10^{10}$
Como 2^{-p} es despreciable frente a la unidad podemos aproximar por: $2^{Emax} > 10^{10}$
Tomando logaritmos: Emax · log2 > 10 → Emax > 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.