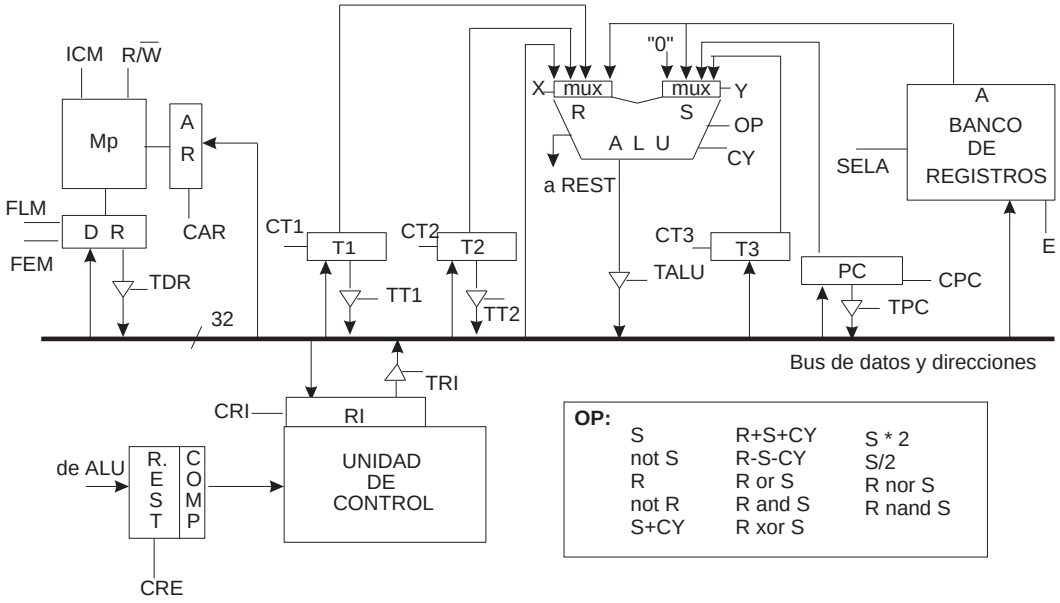
1 *(3,5 puntos) Se muestra en la siguiente figura la estructura de un procesador con palabras y direcciones de 32 bits, un único bus interno, unidad de control cableada y tres registros transparentes: T1, T2 y T3. La memoria tiene un tiempo de acceso de 2 ciclos de reloj y el direccionamiento es a nivel de byte. Los incrementos o decrementos de los registros, se realizan a través de la ALU.*



a) *(80 %) Exprese a nivel RT (transferencia entre registros) las operaciones elementales que se producen, en cada ciclo de reloj, durante la ejecución de cada una de las siguientes instrucciones de una palabra. Incluya la fase de fetch.*

LD .R1, #8[.R2]

ST #8, [.R4]

b) *(20 %) Considerando que la operación elemental de mayor duración tarda un tiempo de 5 ns en realizarse, determine justificadamente el tiempo que tardarían en ejecutarse cada una de las instrucciones anteriores. Debe considerar todas las fases de cada instrucción.*

SOLUCIÓN

a) En primer lugar se indican las operaciones elementales del *fetch* y el ciclo de decodificación, puesto que son idénticas para las dos instrucciones propuestas:

1:AR ← PC; dirección de la instrucción
2:lectura, PC ← PC + 4
3:lectura, DR ← M(AR); carga del registro de datos
4:IR ← DR; carga del registro de instrucción
5:Decodificación

Seguidamente se muestran las operaciones elementales de la fase de ejecución, para cada una de las instrucciones:

LD .R1, #8[.R2]

6:T2 ← IR(desplazamiento)
7:AR ← R2+T2; dirección de lectura
8:lectura
9:lectura, DR ← M(AR)
10:R1 ← DR

```
     ST #8, [.R4]
```

6:AR ← R4; dirección de escritura
7:DR ← IR(inmediato)
8:escritura
9:escritura, M(AR) ← DR

b) La operación elemental de mayor duración determina el mínimo periodo de reloj del procesador, por lo tanto:

$t_{CK} = 5ns$

siendo los tiempos de ejecución de las dos instrucciones:

$t_{ejec} = t_{ciclo} * n^o deciclos$

$t_{ejec_{LD}} = 5ns/ciclo * 10ciclos = 50ns$

$t_{ejec_{ST}} = 5ns/ciclo * 9ciclos = 45ns$

2 *(4 puntos) Programe en ensamblador del 88110 las siguientes subrutinas en las que* **todos los parámetros se pasan en la pila***:*

a) *(50 %)* **VEC(n, V1, V2)** *que compara dos vectores del mismo tamaño y, si son iguales, pone a 0 todos los elementos del vector V2. Cada vector contiene n elementos enteros y los parámetros V1 y V2 son las direcciones de comienzo de los vectores. En esta subrutina, no es necesario que cree el marco de pila.*

b) *(50 %)* **COMP(m, n, Vector, Matriz)** *que compara un vector de n elementos enteros con cada una de las filas de una matriz de tamaño m x n. La matriz está almacenada en memoria por filas y, en aquellas que son iguales al vector, se sustituirá cada uno de sus elementos por 0. Para ello se guardará en la pila, como variable local, un puntero a la fila que se va a comparar y se llamará a la subrutina VEC del apartado anterior. Los parámetros Vector y Matriz son, respectivamente, la dirección de comienzo del vector y la dirección del primer elemento de la matriz, m es su número de filas y n el de columnas.*
En esta subrutina debe crear el marco de pila teniendo en cuenta que la pila crece hacia direcciones decrecientes de memoria y que el puntero de pila apunta a la última posición ocupada. Puede utilizar las macros que considere de las definidas en clase y, tenga en cuenta, que el programa llamante deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP).

SOLUCIÓN

a) La subrutina `VEC` recibe en la pila, como parámetros de entrada, n número de elementos de ambos vectores, V1 dirección de un vector y V2 dirección del otro vector. Trás leer los parámetros y comprobar si los vectores son vacíos, se recorren ambos comparando sus correspondientes elementos. Si los vectores son iguales, tendrá que recorrerse de nuevo V2 para ponerlo a cero.

```
     VEC: ld r2, r30, r0; n
          ld r20, r30, 4; puntero a V1
          ld r21, r30, 8; puntero a V2
          cmp r7, r2, r0
          bb1 eq, r7, final
comparar: ld r5, r20, r0; elemento de V1
          ld r6, r21, r0; elemento de V2
          cmp r7, r5, r6
          bb0 eq, r7, final; distintos
          subu r2, r2, 1; decrementar contador
          cmp r7, r2, r0
          bb1 eq, r7, ceros; iguales
          addu r20, r20, 4; avanzar punteros
          addu r21, r21, 4
          br comparar
   ceros: ld r2, r30, r0; n
```

```
        ld r21, r30, 8; V2
 bucle: st r0, r21, r0; elemento a 0
        subu r2, r2, 1; decrementar contador
        cmp r7, r2, r0
        bb1 eq, r7, final
        addu r21, r21, 4; avanzar puntero
        br bucle
final:  jmp(r1)
```

b) En la subrutina `COMP` se crea el marco de pila reservando una palabra para la variable local. Se llama a la subrutina del apartado anterior para poner a 0 las filas que sean iguales al vector.

```
 COMP: PUSH(r1)
       PUSH(r31)
       or r31, r30, r30    ; crear marco de pila
       subu r30, r30, 4    ; espacio variable local
       ; leer parámetros
       ld r21, r31, 20     ; Matriz
       ld r3, r31, 8       ; m
bucle: ld r20, r31, 16     ; Vector
       ld r2, r31, 12      ; n
       mulu r4, r2, 4      ; n en bytes
       st r21, r31, -4     ; salvar puntero
       ; pasar parámetros
       PUSH(r21)
       PUSH(r20)
       PUSH(r2)
       bsr VEC
       addu r30, r30, 12   ; saltar parámetros
       subu r3, r3, 1
       cmp r7, r3, r0
       bb1 eq, r7, salir
       ld r21, r31 -4      ; recuperar puntero
       addu r21, r21, r4   ; puntero a fila
       br bucle
salir: or r30, r31, r31    ; destruir marco de pila
       POP(r31)
       POP(r1)
       jmp(r1)
```

3 *(2,5 puntos) En un computador con palabras y direcciones de 64 bits y direccionamiento a nivel de byte se dispone de una memoria cache de 64KBytes con bloques de 32Bytes, inicialmente vacía, cuya ubicación es asociativa y su política de escritura WTWNA (inmediata sin actualización). El siguiente fragmento de código está almacenado a partir de la dirección 0 y todas sus instrucciones son de una palabra. Los desplazamientos de las instrucciones de bifurcación se indican en palabras y todos los valores se expresan en decimal.*

```
AND .R5, #0
LD  .R6, #500
LD  .R9, #1024
LD  .R8, [.R9++]
AND .R8, #2047
MOVE.R8, R10
CMP .R5, .R6
BZ  £2
ADD .R5, #1
BR  £-7
WAIT
```

a) *(50 %) Obtenga la traza de referencias a memoria en la ejecución de las cinco primeras iteraciones y determine justificadamente el número de fallos de memoria cache, tanto en los accesos a instrucciones como a datos, para la ejecución completa del código.*

b) *(50 %) Teniendo en cuenta que los tiempos de acceso de la memoria cache y de la memoria principal son, respectivamente, 3 ns y 40 ns, calcule la tasa de acierto en memoria cache y el tiempo efectivo de los accesos a memoria.*

SOLUCIÓN

a) Ya que las palabras son de 64 bits (8 bytes) y los bloques de 32 bytes, cada bloque es de 4 palabras. Teniendo en cuenta que la primera instrucción del programa está contenida en la dirección 0 de memoria, para las primeras cinco iteraciones se tendrá la siguiente traza:

0 8 16 24 1024 32 40 48 56 64 72
24 1032 32 40 48 56 64 72
24 1040 32 40 48 56 64 72
24 1048 32 40 48 56 64 72
24 1056 32 40 48 56 64 72

Para la ejecución completa se tienen 4010 accesos a memoria (3+500.8+7 accesos).

Con respecto a los fallos de cache, se producen 3 fallos en instrucciones (direcciones 0, 32 y 64) y 126 fallos en datos, con lo que el número total de fallos es de 129.

b) La tasa de aciertos en cache y el tiempo de acceso efectivo a memoria son los siguientes:

$Hr_{Mca} = (N^o de\ accesos - N^o de\ fallos)/N^o de\ accesos = 3881/4010 = 0,967$

$t_{ef} = Hr_{Mca} * t_{acierto} + (1 - Hr_{Mca}) * t_{fallo}$

Siendo el tiempo de fallo:

$t_{fallo} = (3 + 4 * 40 + 3)ns = 166ns$

$t_{ef} = 0,967 * 3ns + 0,033 * 166ns = 8,379ns$