

E.P.S.- UAM.- Arquitectura e Ingeniería de Computadores, 21-Noviembre-2008

C1 (2 ptos)	C2 (3 ptos)	C3 (5 ptos)	TOTAL

Apellidos, Nombre	G31	G32	G36	G40	AULA:
					POS:

C1.- Responda breve y concisamente a las siguientes preguntas. Extender innecesariamente las respuestas o contestar a lo que no se pregunta se puntuará negativamente.

C1.1. Para un sistema de memoria de tres niveles, se conocen las tasas de fallos locales (f_{Li}) y los tiempos de acceso a cada nivel (t_i). Indique cuál es la fórmula de tiempo de acceso medio a memoria.

$$\overline{t_{acc}} = t_1 + f_{L1} \cdot (t_2 + f_{L2} \cdot t_3)$$

C1.2. Si se puede elegir entre predicción dinámica de saltos en 1 ó 2 bits, ¿cuál elegiría para código con bucles y por qué?

Predicción dinámica en 2 bits. En cada bucle todos los saltos menos el último son efectivos. Con ambos esquemas de predicción se fallará en el último salto. Sin embargo, con predicción de 1 bit también se fallará en el primer salto del siguiente bucle (habrá cambiado la predicción a no efectiva por el fallo), mientras que en predicción con bits no habrá cambiado por un solo fallo.

C1.3. En cierto sistema el 40% del tiempo se emplea en acceso a memoria y el 30% en operaciones en coma flotante. El sistema de memoria se mejora para reducir a la mitad el tiempo de acceso, y las operaciones en coma flotante se aceleran en un factor de 4. ¿Cuál es la aceleración global? Además del resultado numérico, ponga las fórmulas/desarrollo utilizados.

Mejora 1: memoria. $A_{m1}=2$, $F_{m1}=0,4$.

Mejora 2: coma flotante. $A_{m2}=4$, $F_{m2}=0,3$.

$$Ag = 1 / (1 - F_{m1} - F_{m2} + F_{m1}/A_{m1} + F_{m2}/A_{m2}) = 1 / (1 - 0,4 - 0,3 + 0,4/2 + 0,3/4) = 1,739$$

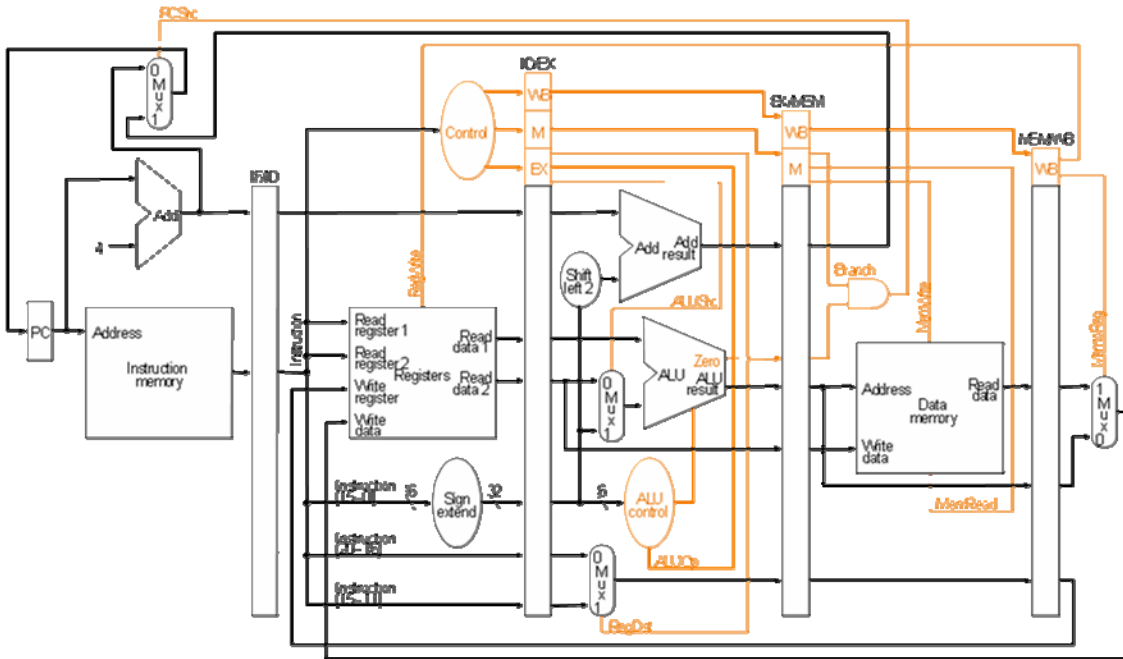
C1.4. En un micro segmentado con arquitectura Harvard el CPI es 2,4 y el porcentaje de cada tipo de instrucción el siguiente: 10% saltos, 60% operaciones aritméticas o lógicas, 20% cargas (*loads*) y 10% almacenamientos (*stores*). Se amplía la caché de datos (sin tocar la de instrucciones) de tal forma que el porcentaje de fallos baja del 10% al 5%, siendo la penalización por fallo de 20 ciclos. ¿Cuál es el nuevo CPI? Además del resultado numérico, ponga las fórmulas/desarrollo utilizados.

Los ciclos de demora que se introducía antes de la mejora por la caché de datos en cada acceso a memoria de datos eran $(1-H) \cdot t_B = 0,1 \cdot 20 = 2$, mientras que tras la mejora son $(1-H) \cdot t_B = 0,05 \cdot 20 = 1$. Por tanto, la mejora en cada acceso a memoria de datos es de 1 ciclo.

Como sólo acceden a memoria de datos las instrucciones *load* y *store*, nos ahorramos 1 ciclo en el 30% de los casos => mejora de $0,3 \cdot 1 = 0,3$ en el CPI.

Nuevo CPI: $2,4 - 0,3 = 2,1$.

C2.- Dado el procesador MIPS estudiado en clase:



La unidad de control se puede implementar en VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
```

```
entity control is
  port(
    opcode : in STD_LOGIC_VECTOR(5 downto 0);
    I_Rdy : in STD_LOGIC; -- No se usa en esta versión
    D_Rdy : in STD_LOGIC; -- No se usa en esta versión
    ALUSrc : out STD_LOGIC;
    ALUOp : out STD_LOGIC_VECTOR(1 downto 0);
    RegDst : out STD_LOGIC;
    BrCond : out STD_LOGIC;
    MemRead : out STD_LOGIC;
    MemWrite : out STD_LOGIC;
    MemToReg : out STD_LOGIC;
    RegWrite : out STD_LOGIC
  );
end control;
```

```
architecture control_arq of control is
begin
```

```
ALUSrc <= '1' when (opcode="100011" or opcode="101011" or opcode="001111")
  -- lw, sw y lui usan el dato inmediato
  else '0';
```

```
ALUOp <= "00" when (opcode="100011" or opcode="101011") -- lw y sw
  else "01" when (opcode="000100") -- beq
  else "10" when (opcode="000000") -- add, sub, and, or, slt
  else "11" when (opcode="001111") -- lui
  else "00"; -- j e instrucciones no validas
```

```
RegDst <= '1' when (opcode="000000") -- add, sub, and, or, slt
  else '0'; -- resto de instrucciones no tienen 3 registros
```

```
BrCond <= '1' when (opcode="000100") -- beq
  else '0'; -- resto de instrucciones
```

```
MemRead <= '1' when (opcode="100011") -- lw
  else '0'; -- resto de instrucciones
```

```
MemWrite <= '1' when (opcode="101011") -- sw
  else '0'; -- resto de instrucciones
```

```
MemToReg <= '1' when (opcode="100011") -- lw
  else '0'; -- resto de instrucciones
```

```
RegWrite <= '1' when (opcode="100011" or opcode="000000" or opcode="001111")
  -- escriben en registro lw, add, sub, and, or, slt y lui
  else '0'; -- resto de instrucciones
```

```
end control_arq;
```

E.P.S.- UAM.- Arquitectura e Ingeniería de Computadores, 21-Noviembre-2008

Si se ejecutan las instrucciones:

```
lw $t1, 24($zero) # lw $r9, 24($r0)
lw $t2, 28($zero) # lw $r10, 28($r0)
add $t7, $t1, $t3 # add $r15, $r9, $r11 => r15=r9+r11
and $s0, $t1, $t2 # and $r16, $r9, $r10 => r16=r9 and r10
sw $t7, 40($zero) # sw $r10, 40($r0)
sw $s0, 44($zero) # sw $r11, 44($r0)
```

Se pide completar la tabla a partir del momento en que se decodifica el segundo load (lw).

	T:lw \$t1	T+1	T+2	T+3	T+4	T+5	T+6	T+7	T+8
ALUSrc	1	1	0	0	1	1			
ALUOp	00	00	10	10	00	00			
RegDst	0	0	1	1	X	X			
BrCond	0	0	0	0	0	0			
MemRead	1	1	0	0	0	0			
MemWrite	0	0	0	0	1	1			
MemToReg	1	1	0	0	0	0			
RegWrite	1	1	1	1	0	0			
IDEX_ALUSrc	-	1	1	0	0	1	1		
IDEX_ALUOp	-	00	00	10	10	00	00		
IDEX_RegDst	-	0	0	1	1	X	X		
IDEX_BrCond	-	0	0	0	0	0	0		
IDEX_MemRead	-	1	1	0	0	0	0		
IDEX_MemWrite	-	0	0	0	0	1	1		
IDEX_MemToReg	-	1	1	0	0	0	0		
IDEX_RegWrite	-	1	1	1	1	0	0		
EXMEM_BrCond	-	-	0	0	0	0	0	0	
EXMEM_MemRead	-	-	1	1	0	0	0	0	
EXMEM_MemWrite	-	-	0	0	0	0	1	1	
EXMEM_MemToReg	-	-	1	1	0	0	0	0	
EXMEM_RegWrite	-	-	1	1	1	1	0	0	
MEMWR_MemToReg	-	-	-	1	1	0	0	0	0
MEMWR_RegWrite	-	-	-	1	1	1	1	0	0

Donde el primer grupo de señales son las que genera la unidad de control, las que tienen el prefijo IDEX son las señales de control registradas en el registro ID/EX de la figura, las que tienen el prefijo EXMEM son las señales de control registradas en el registro EX/MEM de la figura, y las que tienen el prefijo MEMWB son las señales de control registradas en el registro MEM/WR de la figura.

Nota: Suponga que los adelantamientos de datos están completamente resueltos en hardware.

E.P.S.- UAM.- Arquitectura e Ingeniería de Computadores, 21-Noviembre-2008

C3.- El ordenador educativo *ARQUI01* dispone de un procesador cuyas direcciones virtuales son de 20 bits y dispone de una memoria real de 64 Kbytes. Posee un sistema de memoria paginado, con un tamaño de página de 256 bytes y longitud del descriptor de página de 4 bytes en el que se incluye el marco de página, bit de presencia y varios bits de control. Como la tabla de páginas es muy grande, ésta a su vez está paginada en dos niveles (el primero siempre está presente en la memoria principal). La unidad de manejo de memoria (MMU) del sistema dispone de un TLB completamente asociativo de 8 entradas. El tamaño de la ruta de datos y, por tanto, el tamaño de las lecturas es de 16 bits. El sistema posee además caché virtual unificada asociativa de 4 vías, 16 bloques en total y 2 palabras por bloque.

En las figuras se representa el contenido del TLB, el contenido de la caché y de algunas posiciones de memoria relevantes para el problema. Los datos están organizados "little endian" (byte de menor peso en dirección más baja). La información del marco está en los bits de menor peso de las 2 palabras. Suponer que la página que se busca está presente. Se pide:

a) Cómo se divide la dirección para el acceso a caché, TLB y memoria (rellenar las plantillas). Indicar el tamaño del TLB y de la caché. Expresar tanto el tamaño de datos como el tamaño total y cantidad de comparadores en el caso de la memoria caché.

b) Indicar los pasos que se han de seguir cuando el procesador genere las direcciones E2C58₁₆, CC2BC₁₆ y 1C2B2₁₆ y qué resultado se devolverá al procesador.

c) Indique (en la tabla adjunta de la próxima página) en qué posiciones de la memoria caché habrá actualizaciones para los accesos a memoria del apartado b). Suponga que la vía LRU para todos los casos es la vía 4. No es necesario indicar en las tablas la actualización de la caché, sólo en qué posiciones.

Contenido del TLB

Etq.	Ctrl	MPR
245	X...X	CE
F56	X...X	22
E2C	X...X	01
001	X...X	2E
11D	X...X	01
CC2	X...X	2B
344	X...X	D4
AB1	X...X	ED

Caché Virtual Unificada

Vía 1		Vía 2		Vía 3		Vía 4	
Etq	Datos	Etq	Datos	Etq	Datos	Etq	Datos
0	CCCC ABCD EF12	CCCD	ABDC EF21	CDCC	ABCE EF13	CCFC	ABEC EF31
1	CCCC ABCD EF14	CCCD	ABDC EF23	CDCC	ABCE EF15	CCFC	ABEC EF33
2	A132 2007 2006	E2C5	2008 2009	31B2	EABC 1ECD	1132	ABCD EF12
3	3B21 5E83 197F	1234	FA19 38B4	1C2B	29D3 8930	13B2	3830 5740

Base 1er nivel de páginas: 5500 hex

Contenido de la Memoria Principal

Pos(hex)	Val(H)
0000	XX
...	...
0100	12
0101	13
0102	14
0103	24
...	...
0158	09
0159	20
015A	08
015B	20
...	...
13D4	E0
13D5	E3
...	...
2BBC	A0

Pos(hex)	Val(H)
2BBD	A2
2BBE	AA
2BBF	22
...	...
3B2E	31
3B2F	32
3B30	55
3B31	CF
3B32	32
...	...
551C	CC
551D	22
551E	24
551F	DD
...	...
5C49	CC

Pos(hex)	Val(H)
5C4A	CC
5C4B	DD
...	...
810F	45
8110	67
8111	88
8112	98
...	...
C857	01
C858	33
...	...
CC08	E0
CC09	E1
...	...
CDB4	20
CDB5	05

Pos(hex)	Val(H)
CDB6	56
CDB7	78
...	...
DE09	40
DE0A	20
...	...
E0B0	21
E0B1	EE
E0B2	D0
E0B3	DE
...	...
F1AB	BB
F1AC	DE
...	...
FFFE	XX
FFFF	XX

E.P.S.- UAM.- Arquitectura e Ingeniería de Computadores, 21-Noviembre-2008

a)

Dir. virtual para acceso a tablas

Nivel 1	Nivel 2	Offset
6 bits	6 bits	8 bits

Dirección real

MPR	Offset
8 bits	8 bits

MPR: Marco de Pág. Real

Campos del TLB

Etiqueta	Control	MPR
12 bits	24 bits	8 bits

Tamaño de campos del descriptor

Control	MPR
24 bits	8 bits

Campos de la dirección virtual para acceso a la caché

Etiqueta	Índice	B/B
16 bits	2 bits	2 bits

Breve justificación y tamaños TLB y caché:

DV: 20 bits; DR: 16 bits; Páginas: 256 Bytes = 2^8 Bytes => 8 bits de offset

En las tablas de páginas alojadas en páginas se pueden guardar $256/4 = 64 = 2^6$ descriptores

Si la tabla tiene 2 niveles: 20 (total) = 6 (nivel1) + 6 (nivel2) + 8 (offset)

Acceso a la caché 20 bits; 2 Byte in Block (2 palabras de 2 bytes cada una); 2 bits de índice ya que hay 4 bloques por vía (16 bloques/4 vías), Etiqueta: $20 - 2 - 2 = 16$ bits

Tamaño total TLB: $8 \times (12 \text{ bits etiquetas} + 32 \text{ bits descriptor}) = 352 \text{ bits} = 44 \text{ Bytes}$

Tamaño caché, parte de datos: 4 vías x 4 bloques/vía x 2 palabras/bloque x 2 Bytes/pal = 64 Bytes

Tamaño total caché: 4 vías x 4 bloques/vía x (2 Bytes de etiqueta + 4 Bytes datos) = 96 Bytes

Comparadores caché: 4 comparadores de 16 bits

b) Indicar los pasos que se han de seguir cuando el procesador genere las direcciones $E2C58_{16}$, $CC2BC_{16}$ y $1C2B2_{16}$ y qué resultado se devolverá al procesador.

b.1) Dir Virtual: $E2C58_{16}$; Búsqueda en caché: Etiq: $E2C5$ con índice 10_2 , acierto en vía 2; Dato Obtenido: 2009_{16} ya que el B/B es 00 (parte baja).

b.2) Dir Virtual: $CC2BC_{16}$; Índice para acceso a caché = 11_2 ; Fallo en caché;

Acceso al TLB: Etiqueta: $CC2$; acierto en TLB, marco 2B; luego la $DR=2BBC$

Dato Obtenido: **A2A0**₁₆ Direcciones (2BBD y 2BBC). Se carga en caché entrada 11_2 las posiciones de memoria 2BBF-2BBE-2BBD-2BBC (22AA A2A0) con etiqueta $CC2B$

b.3) Dir Virtual: $1C2B2$; índice para acceso a caché: = 00_2 ; Fallo en caché;

Acceso al TLB: Etiqueta: $1C2$; Fallo en TLB

Búsqueda en la tabla: #pág en binario 0001 1100 0010 => índice $L1 = 000111_2 = 7_{16}$; índice $L2 = 000010_2 = 2_{16}$;

Luego si la base de página es 5500, la entrada en la tabla de 1er nivel es: $5500_{16} + 7_{16} * 4 = 551C_{16}$.

Contenido DD2422CC. Es decir, el marco es CC.

El acceso al segundo nivel es $CC00_{16} + 2_{16} * 4 = CC08_{16}$. Siendo el marco buscado E0. Luego $DR=E0B2$.

Dato Obtenido: **DED0**₁₆ Direcciones (E0B3 y E0B2). Se actualiza caché con las direcciones (E0B3, E0B2 E0B1 y E0B0). DED0 EE21, etiqueta $1C2B$ y TLB con etiqueta $1C2$ y marco E0.

c) Indique (en la tabla adjunta a continuación) en qué posiciones de la memoria caché habrá actualizaciones para los accesos a memoria del apartado b). Suponga que la vía LRU para todos los casos es la vía 4. No es necesario indicar en las tablas la actualización de la caché, sólo en qué posiciones.

	Vía 1		Vía 2		Vía 3		Vía 4	
	Etiqu	Datos	Etiqu	Datos	Etiqu	Datos	Etiqu	Datos
0							1C2B	DED0 EE21
1								
2								
3							CC2B	22AA A2A0

b.1 No modifica caché.

b.2. Se carga en caché entrada 11_2 las posiciones de memoria 2BBF-2BBE-2BBD-2BBC (22AA A2A0), como la LRU es la vía 4 se coloca en ella. La etiqueta es $CC2B$.

b.3. Se actualiza caché con las direcciones (E0B3, E0B2 E0B1 y E0B0). DED0 EE21, etiqueta $1C2B$.