66.20 Organización de Computadoras
Resolución de ejercicios posibles de Final
Facultad de Ingeniería, Universidad de Buenos Aires

Martín E. Buchwald
Ezequiel Genender Peña
(Colaboración de Nicolás Menzano Díaz, con su resumen de las clases)

Índice

1. Temas de Arquitecturas

1) Describa mediante ejemplos los modos de direccionamieto que conozca

Hacemos un ejemplo con un simple *add*:

- Por registro: *add r3, r4, r5* (r3 = r4 + r5, en todos casos, los valores en los registros directamente)
- Inmediato: *add r1, r1, #5* (r1 = r1 + 5)
- Desplazamiento: *add r1, 100(r4)* (a r1 le sumo el valor de la posicion de memoria del valor guardado en r4 + 100 posiciones).
- Indirecto por registro: *add r1, (r2)* (idem desplazamiento pero sin offset)
- Indexado: *add r4, (r1+r2)* (Se le suma a r4 el valor en la pos de memoria r1+r2)
- Directo o Absoluto: *add r4, (1000)*
- Indirecto por memoria: *add r4, @(r3)* Esto quiere decir, que a r4 se le sumara el valor encontrado en la posicion apuntada por la posicion apuntada por r3 (ej, en r3 tengo 0x40000, y en la posicion de memoria 0x40000 tengo 0x32000, entonces a r4 le sumo el valor de la posicion 0x32000)
- Autoincremento: *add r4, (R2)+* Accede a la posición de R2, y R2 se autoincrementa para acceder a la próxima dirección de memoria
- Escalado: *Add r4, 100(R2)[R3]*, se le suma a r4 el valor en la posicion: r2 + 100 posiciones contiguas, y R3 posiciones más adelante.

En particular en MIPS, solo contamos con: Por registro e inmediato, y podemos usar el de desplazamiento para accesos a memoria únicamente (*lw* y *st*) (podría ser lo mismo con directo también).

2) Describa los distintos tipos de arquitecturas de conjunto de instrucciones que conozca. De sus ventajas y desventajas.

Arquitectura Stack: Todos los operandos deben pasar de memoria al Stack. La ALU hace operaciones con los dos elementos superiores de la pila, desapilandolos y apilando el resultado. Ventaja: en si para cuestiones de set de instrucciones, al no haber demasiadas formas de hacer las cosas, suele ser bastante sencillo, sin demasiadas instrucciones. Además, el hardware no debería ser demasiado complicado, porque siempre saca los operandos del mismo lugar, y pone los resultados en el mismo lugar. Desventaja: primero, es un bodrio tener que pasar todo al stack por cada cuanta que se quiera hacer, y sacar el resultado también, entonces el código termina siendo muy largo, lleno de "PUSH/POP", además, está claro que tiene la dificultad de tener que acceder constantemente a memoria para todo, lo que genera que sea una arquitectura lenta.

Arquitectura Acumulador: Contamos con único registro, el *acumulador*. Un operando de la ALU es siempre el acumulador, y el otro proviene de la memoria. El resultado se guarda

en el acumulador. Ventaja: nuevamente, el código a escribir no tiene muchas maneras de ser escrito, entonces suele ser simple la solución, y otra vez, el hardware no tiene que ser demasiado complejo (una entrada y la salida serán siempre los mismos, y sólo es necesario indicar en dónde está el segundo operando). Desventaja: otra vez, hay que pasar siempre por la memoria.

Estas dos arquitecturas (stack - acumulador) serán realmente limitadas para resolver problemas de mayor escala.

Arquitectura Registro - Memoria: En esta arquitectura, hay varios registros de propósitos generales. Uno de los operandos serán algún registro, y el otro proveniente de la memoria. Además, el resultado se guardará en el registro origen. Ventaja: No es necesario cargar todos los operandos desde memoria, y tiende ser fácilmente codificado. Desventaja: Los operandos no se tratan de la misma manera (el que está guardado en el registro desaparece). Además, suelen quedar pocos bits para codificar el registro, por lo que suelen haber pocos.

Arquitectura Registro - Registro: Se cuenta con varios registros, a tal punto que se puede destinar a distintos usos para utilizar nomenclaturas a la hora de programar. Todas las operaciones se hacen con operadores registros, a un destino registro. Ventaja: por esto último, solo es necesario acceder a memoria para obtener o guardar un dato (por esto, a veces se la suele llamar arquitectura Load-Store), por lo que se elimina un gran problema de cuello de botella. Además, las instrucciones pueden tener un largo fijo, lo que permite una sistematización de circuitería mucho más simple, y las instrucciones suelen tardar una cantidad similar de ciclos de clock (y las que más tardan, son las de acceso a memoria). Desventaja: Los programas tienden a ser bastante grandes, por lo que ocuparán más espacio (pues hay que tener instrucciones para traer y llevar datos desde y hacia memoria).

Arquitectura Memoria - Memoria. Lo mismo de siempre, pero los operandos siempre pasan por memoria, y se guardan a memoria. Ventaja: Código compacto (no es necesario cargar los operandos), y no hay dependencia con registros. Desventaja: Las instrucciones varian en tamaño, y el CPI suele ser muy variable, y obviamente está el problema de estar todo el tiempo accediendo a memoria.

3) Codifique la línea de código $A = B + C$ para las arquitecturas:
a) Stack
b) Acumulador
c) Load-Store
d) Memoria-Memoria.
Explique brevemente las ventajas y desventajas de cada una de ellas.

a) Las ventajas y desventajas ya fueron explicadas en el punto 2.

```
PUSH B
PUSH C
ADD
POP A
```

b) Las ventajas y desventajas ya fueron explicadas en el punto 2.

LOAD B
ADD C
STORE A

c) Idem.
LOAD R1, B
LOAD R2, C
ADD R3, R1, R2
STORE A, R3

d) Idem.
ADD A, B, C

Quedando claro que en la arquitectura Stack queda todo lleno de "Push/Pop", en la de acumulador la cosa suele ser simple, pero limitada, en la Load-Store se necesitan más líneas de código, pero para realizar la operación no es necesario acceder a memoria, y en la Memoria-Memoria, la cantidad de líneas es mínima (pero para la misma instrucción se accede 3 veces a memoria).

4)Entrada-Salida: Que se entiende para una arquitectura en relación a la entrada-salida que ésta sea de un bus o de dos buses?
Falta esta respuesta!

5)Entrada-Salida: Explicar claramente que es Entrada-Salida mapeada en memoria. Conoce algún otro esquema opuesto al anterior? Que características tiene?

Que los periféricos de entrada-salida estén mapeados en memoria, implica que desde el programa podemos acceder a ellos como un simple espacio en memoria cualquiera, siendo esta parte de la memoria principal, por lo que, en sí, el hecho de que sea un espacio de memoria o un periférico es "transparente" al programa (hasta ahí, pues se indica una posición fija por algo).
Existe, por otro lado, el caso de tener un Mapeo externo, lo que quiere decir que los dispositivos tienen su propio espacio de direcciones. Por esto, no podemos tratarlo como un espacio de memoria cualquiera, sino que, para direccionar a un dispositivo cualquiera, es necesario tener un set de instrucciones más grande, para poder elegir entre la memoria principal, o el de I/O.

6) Para el procesador MIPS indicar si los siguientes elementos están determinados por la arquitectura del conjunto de instrucciones o por la implementación. Justificar.
a) Superescalar.
b) Espacio de direcciones virtuales.
c) Espacio de direcciones físicas.
d) Pipeline de 5 etapas.
e) Unidad de control microprogramada.
f) Inmediatos de 16 bits.
g) Cache de nivel 1.

a) El hecho de que el sistema sea superescalar es totalmente una cuestión de implementación, porque es la decisión de procesar dos operaciones simultáneamente. Esto no está determinado por el set de instrucciones, pues justamente para el programador, el código se va ejecutando linealmente.

b) El espacio de direcciones virtuales sí está determinado por el set de instrucciones (por algo se llama MIPS32, no?), pues para el set de instrucciones, el programa tiene a mano una cantidad de memoria especificada de antemano, e independientemente de la implementación que tengamos. Justamente por esto, podemos acceder a cualquier posición (dentro del rango determinado, y sea permitido) sin incurrir en errores.

c) A diferencia de b), justamente esto es dependiente de la implementación. El punto b) está para subsanar este punto, pues uno puede determinar la memoria física que uno quiera tener (incluso más que la virtual, pues se puede aprovechar el uso de la memoria virtual para procesamiento multihilo), lo cual debe ser totalmente transparente a un programa.

d) La decisión de utilizar Pipeline de 5, 4, media etapa es puramente de implementación. Para un mismo set de instrucciones puede implementarse o no el Pipeline. La utilización de Pipeline es para aprovechar el paralelismo de las instrucciones, aplicando el consepto de línea de montaje. La implementación del Pipeline es totalmente transparente al set de instrucciones, y al programador.

e) Esto no creo que lo tome más. REVEER, porque no estoy totalmente seguro (creo que la respuesta es que depende del set de instrucciones, pues se aprovecha que, por ejemplo, el opcode se encuentra siempre en el mismo lugar).

f) Esto depende del set de instrucciones. Si fuera por la implementación en sí (teniendo que adaptarla, claro), se podrían sumar 32 bits (o 64, u otro valor, dependiendo del set de instrucciones) sin ningún problema, pero es imposible dentro de los 32 bits de la instruccion entren tal constante y la información de la operación. Se pueden utilizar 16 bits porque es lo que entra para seguir definiendo el opcode, el primer operador y el destino.

g) Justamente la Caché (sea del nivel que sea, o de la forma que sea) conceptualmente, es transparente al programador, y por ende al set de instruccciones. Por ende, es determinada por la implementación.

7) Indique algunas de las características adicionales a las presentes en un procesador de alto desempeño que pueden encontrarse y son específicas de un procesador DSP.

La arquitectura que implementa un DSP es Harvard, no Von Neumann porque está pensado para un fin específico: conversiones analógico-digitales utilizando métodos numéricos para los cálculos. Sirve para representar señales analógicas en tiempo real (podría usarlo un ORC, pero lo dudo...).
No hay más información que esta en la wiki.

8) Para el procesador MIPS indicar si los siguientes elementos están determinados por la arquitectura del conjunto de instrucciones o por la implementación. Justificar.
a) Tamaño de la instrucción.
b) Espacio de direcciones virtuales.
c) Espacio de direcciones físicas.
d) Pipeline de 5 etapas.
e) Unidad de control microprogramada.
f) Conjunto de instrucciones.
g) Cache de nivel 1.

Idem 6 excepto el a). Este depende del set de instrucciones, pues justamente, dependiendo de esta, la instrucción tendrá un tamaño distinto. Incluso, dependiendo de la arquitectura del set de instrucciones, las instrucciones pueden tener tamaño variable.

9) Describa los tres formatos de instrucción MIPS, indicando nombre, tamaño y finalidad de cada uno de los campos. Ejemplifique con una instrucción por formato, en ensamblador y en binario.

TU VIEJA va a perder tiempo con un ejercicio así. Es simplemente un ejercicio de memoria, no de cuánto sabés de la materia. Igualmente, ya que estamos, lo respondemos. Tenemos:

- Instrucciones de tipo R (registro - registro). Son operaciones de ALU directamente. Se cuenta (de más significativo a menos) 6 bits de OPCODE, 5 de rs (register source), 5 de rt (register tsource (?)), 5 de rd (register destiny), 5 de shamt (se usa para indicar la cantidad de lugares a mover en un shift) y 6 de function (se usa para indicar el tipo de operación). (rd = rd function rt). Ejemplo add r3, r1, r2. No me se ni el opcode (creo que es 0 para todas estas operaciones), ni el function del add, pero shamt = 0, rs = 00001, rt = 00010, rd = 00011.

- Instrucciones de tipo I (de tipo inmediatas). Son las operaciones de acceso a memoria, y branch condicionales. Se tiene, como siempre, 6 bits de OPCODE, 5 bits de rs, 5 bits de rt, y los 16 bits restantes, para un inmediato. Cada operación de tipo inmediata tiene un opcode distinto (no como en el caso de las de ALU, subu y addu tienen mismo opcode, pero no addui y subui). Ejemplo: lw r3, 4(r1). En este caso, OPCODE ni idea cuanto vale, pero rs = 00001, rt = 00011 y el inmediato = 0000000000000100.

- Instrucciones de tipo J (jump). Sirven para hacer el jump and link, y otras operaciones similares. Tienen 6 bits de opcode y el resto de los 26 bits son de offset a sumarle al PC. Ej: j 0x000010. Creo que el opcode vale 2, y bueno, el resto de los 26 bits tendran el 0x10.

10- Describa las características principales de una arquitectura load/store. Load-/Store y RISC son sinónimos?
La idea de la arquitectura Load-Store, o Registro-Registro, es limitar los accesos a memoria únicamente a la hora de obtener datos de ella (loads) o guardar datos en ella (store). Para esto justamente, se utilizan muchos registros para poder tener un mejor manejo. Todas las operaciones ALU se realizan entre registros, hacia registros (el resultado). Para ver ventajas

y desventajas, ver Ejercicio 2. Sobre si RISC y load/store son sinónimos, no necesariamente. RISC usa necesariamente una arquitectura Load/Store, pero no porque se use una arquitectura Load/Store será necesariamente una RISC.

11- Para el código MIPS obtenido del siguiente código C:

```
1: void main() {
2:      int cont = 0, i;
3:      for(i = 1;i <= 10;i++) {
4:         if(i % 4 == 0)
5:            cont++;
6:         if(i % 2 == 0)
7:            cont--;
8:      }
9: }
```

```
Direccion    Binario    Codigo leng. Ensamblador
9D000018    27BDFFF0    addiu   sp,sp,-16
9D00001C    AFBE0008    sw      s8,8(sp)
9D000020    03A0F021    addu     s8,sp,zero
9D000024    AFC00000    sw      zero,0(s8)
9D000028    24020001    addiu   v0,zero,1
9D00002C    AFC20004    sw      v0,4(s8)
9D000030    8FC20004    lw      v0,4(s8)
9D000034    2842000B    slti     v0,v0,11
9D000038    10400014    beq     v0,zero,0x9d00008c
9D00003C    00000000    nop
9D000040    8FC20004    lw      v0,4(s8)
9D000044    30420003    andi     v0,v0,0x3
9D000048    14400004    bne     v0,zero,0x9d00005c
9D00004C    00000000    nop
9D000050    8FC20000    lw      v0,0(s8)
9D000054    24420001    addiu   v0,v0,1
9D000058    AFC20000    sw      v0,0(s8)
9D00005C    8FC20004    lw      v0,4(s8)
9D000060    30420001    andi     v0,v0,0x1
9D000064    14400004    bne     v0,zero,0x9d000078
9D000068    00000000    nop
9D00006C    8FC20000    lw      v0,0(s8)
9D000070    2442FFFF    addiu   v0,v0,-1
9D000074    AFC20000    sw      v0,0(s8)
9D000078    8FC20004    lw      v0,4(s8)
9D00007C    24420001    addiu   v0,v0,1
9D000080    AFC20004    sw      v0,4(s8)
9D000084    1000FFEA    beq     zero,zero,0x9d000030
9D000088    00000000    nop
9D00008C    03C0E821    addu     sp,s8,zero
```

```
9D000090    8FBE0008    lw      s8,8(sp)
9D000094    27BD0010    addiu   sp,sp,16
9D000098    03E00008    jr      ra
9D00009C    00000000    nop
```

I- Calcular:
a) El tamaño del código en bytes.
b) La cantidad de instrucciones ejecutadas.
c) La cantidad de referencias a datos en memoria.
d) La cantidad total de bytes transferidos entre CPU y memoria.
e) El número promedio de accesos a memoria por instrucción.
f) El porcentaje de referencias a memoria que son lecturas.
g) El CPI del código.

II -a- Explicar que instrucciones resuelven las operaciones $i\,\%4$ e $i\,\%2$ y como lo hacen.
b- Verdadero-Falso. Justificar:
b1- Las NOPS las inserta el compilador para evitar riesgos tipo RAW.
b2- El código está optimizado.
b3- El op-code de addiu es dec 9.

I) a) Sabemos que cada línea de código (en lenguaje ensamblador) ocupa 32 bits = 4 bytes. En este caso, tenemos 34 líneas de código, por lo que termina ocupando 136 Bytes.
b) Aclaración: Como no se especifica si se trabaja con un esquema de salto demorado, y no puedo estar seguro de si los nops se ejecutan algunas veces o no, simplemente me aferro al enunciado que dice "operaciones" ejecutadas, y digo que el nop no lo es. Sino habría que definir el tipo de esquema, o si contar o no los nops. Igual la idea sería la misma.
Acá tenemos que ir analizando un poco el código. Para ya tener una parte hecha, vemos las porciones de código fuera del loop (son en total 10). Luego vemos cuantas se ejecutan en todas las veces del loop (son en total 13, pero hay que tener en cuenta que el primer branch se ejecuta una onceava vez). Finalmente, contamos cuantas lineas son para el caso que i sea multiplo de dos (cuantas lineas se ejecutan en caso de que $i\,\%2 == 0$) (en total 3), y para el caso de que sea multiplo de cuatro (sin contar la parte de que será multiplo de 2) (en total 3). Por lo tanto, el total de operaciones ejecutadas serán:

$$Op = 10 \times 1 + 13 \times 10 + 1 + 3 \times 5 + 3 \times 2 = 162$$

c) Nuevamente, hay que ver cuántas veces se accede a memoria dependiendo de si se está o no en el Loop.

- Fuera del loop: 4
- Dentro del loop principal (las que se ejecutan siempre): 5
- Si es múltiplo de 2: 2
- Si es múltiplo de 4: 2

Por lo tanto:

$$Accesos = 4 + 5 \times 10 + 2 \times 5 + 2 \times 2 = 68$$

d) Sabiendo la respuesta de c), dado que en cada transferencia de CPU-Memoria hay 32 bits (ya sea de CPU a memoria o viceversa), habiendo un total de 68 accesos a datos, tendrémos un total de 2176 bits de acceso a datos, o sea 544 Bytes.
Además, hay que considerar las transferencias por acceso a instrucciones. Dinámicamente, se ejecutaron 162 instrucciones. A 32 bits por instrucción, se acceden a (162*32) 5184 bits.
En total: 7360

e) Sabiendo b) y c): 0.42 para datos. Como para cada instrucción se hace un acceso para la instrucción misma, resultado = 1.42

f) Habría que hacer c) sólo para lecturas, lo que da un total de 47, por lo que da un 69 %.

g) Si no me decís cuánto tardan los accesos a memoria, y el CPI de ejecución (o si es muy forro, el de cada tipo de instrucción), me va a costar resolverlo. Si lo asumimos de dato:

$$CPI = \frac{\text{Inst. sin acc}}{\text{total inst}} CPI_{ej} + \frac{\text{Acc a mem}}{\text{total inst}} T_{ACC} = 0{,}58 \times CPI_{ej} + 1{,}42 \times T_{ACC}$$

II) a) Las operaciones que resuelve la comparación con módulo son (además de la carga del operando desde el stack) el `and` que se le aplica como máscara, aprovechando que se quiere hacer la operación módulo con elementos de la base 2. Si queremos ver que un número sea múltiplo de 2, sólo tenemos que ver el último bit, si es 1 no lo es, si es 0 si. Para quedarnos únicamente con ese bit, se le aplica una "máscara" con una and con 0x1 (luego se aplica un bne). Lo mismo para 4, pero necesitamos ver que los dos bits menos significativos sean 0, por lo que la máscara a aplicar es 0x3 (00011).

b) 1. Falso. Esto no evita de ninguna manera riesgos RAW (pues la operación antes utilizada es un branch, y no hay escritura allí), sino para evitar los riesgos de control (justamente, por los branch), para el caso en el que haya un esquema de salto demorado (con un delay slot).
2. Si la pregunta viene por el lado de los nops, habría que ver en profundo el código si no es posible reutilizar el delay slot para alguna operación que se realice siempre. Para mi la respuesta es: SI, porque luego de cada uno de los branch (excepto el último), hay una operación de acceso a memoria. Si se reutilizara el delay slot se estaría accediendo a memoria (por más que luego se pisaría el dato, cosa que está bien), se estaría accediendo a memoria, cosa que no está copado (REVISAR!! porque si uno está usando un pipeline, si el ciclo de reloj está ajustado para que labure a la velocidad de la memoria (poco probable) está todo bien, pero hay que tener en cuenta que está el tema de la caché, que el miss y bla bla), y no se pueden realizar las siguientes operaciones porque dependen de ese acceso.
3. Vamos a la primera operación del código, y vemos que es una addiu. El Hexadecimal de la operación es 0x27BDFFF0, en binario: 0010-0111-etc... Como el opcode son los 6 bits más significativos: opcode = 001001 = 9. Por lo tanto, Verdadero.

2. Unidad de Control

1- Unidad de control microprogramada.
Mucho gusto, me llamo Martín....

2- Escribir la secuencia de microoperaciones necesarias para que una de CPU MIPS con microarquitectura de un único bus interno y unidad de control microprogramada ejecute la instrucción LW (load word). A tal efecto dibuje el datapath a utilizar y defina las señales a utilizar explicando su acción.

Tengo entendido que este tema ya no se toma más, y la verdad que sería un bajón importante que lo tome, porque en ningún momento siquiera lo mencionó en alguna teórica... VER SI LUEGO LO PODEMOS HACER

3. Desempeño

1-Escribir la ecuación de desempeño de CPU. Indicar para cada uno de los componentes de la ecuación si éste es afectado o no por:
a) Tecnología del hardware.
b) La organización.
c) La arquitectura del conjunto de instrucciones.
d) Tecnología del compilador.

Ecuación:

$$T_{CPU} = IC \cdot CPI \cdot \tau \tag{1}$$

Siendo IC la cantidad de instrucciones ejecutadas, CPI la cantidad de ciclos por instrucción (promedio en general), y τ el período del clock.
a) Este sólo afecta al clock.
b) Este afecta al CPI (no da lo mismo si usamos un pipeline de 4 etapas a uno de 5, por ejemplo). Llegado el caso, podría afectar al clock, pues habría que ajustarlo a la instrucción que más tarde.
c) Modifica al IC (pues el compilador al pasar a lenguaje ensamblador, dependiendo de la arquitectura habrán más o menos instrucciones a ejecutar), y también al CPI. Ejemplo: arq. Reg-Reg vs. Mem-Mem. En el primero se tienen más instrucciones para un mismo programa, pero menos accesos a memoria, por lo que el CPI tiende a ser más estable para los distintos tipos de instrucciones, mientras que para el segundo el código es más compacto, pero tiene muchos más accesos a memoria, haciendo variar el CPI de distintas instrucciones mucho más, y teniendo en general un CPI promedio mucho más alto.
d) Este afecta simplemente al IC, teniendo en cuenta optimizaciones y esas cosas.

2) Considere la siguiente mezcla de instrucciones:

Operación	Frecuencia	CPI
ALU op	40 %	1
branch no tomado	10 %	1
branch tomado	20 %	4
load/store	30 %	2

Para incrementar el desempeño que sería mejor?
a) reducir el CPI de los branches tomados a 1, o
b) acelerar la velocidad del reloj por un factor de 4/3 (= 1.33)

Para resolver esto, calculamos el Speed Up en cada caso.
a) Recordando la ecuación de desempeño del CPU, y asumiendo que el IC (mismo programa) es el mismo, y el clock también:

$$S_{UP} = \frac{T_V}{T_N} = \frac{CPI_V}{CPI_N} = \frac{0{,}4 \times 1 + 0{,}1 \times 1 + 0{,}2 \times 4 + 0{,}3 \times 2}{0{,}4 \times 1 + 0{,}1 \times 1 + 0{,}2 \times 1 + 0{,}3 \times 2} = \frac{1{,}9}{1{,}3} = 1{,}46$$

b) Acelerando el Clock estamos disminuyendo su período 1.33 veces. Asumiendo CPI e IC

iguales antes y después de la optimización:

$$S_{UP} = \frac{\tau_V}{\tau_N} = \frac{\tau_V}{\frac{1}{1{,}33}\tau_V} = 1{,}33$$

Por lo tanto, la mejor opción es la a).

3) Un conjunto de benchmarks representativo de la carga habitual de un modelo de procesador arroja una distribución en el uso de instrucciones del siguiente tipo:

Operación	Frecuencia
Enteras	81 %
Punto Flotante	19 %

Donde las instrucciones enteras son 10 veces más rápidas que las de punto flotante. Se propone una mejora en hardware de punto flotante de manera tal que las instrucciones de punto de flotante logran una aceleración de 2. Calcular la aceleración global que se obtiene de aplicar dicha mejora. Aclare las hipótesis que utilice.

Asumo por hipótesis que las mejoras implementadas no modifican de ninguna manera cómo se comporta la parte de enteros. Además, asumo que se utiliza el mismo programa y computadora (IC y clock constantes). Entonces:
Dadas las hipótesis, el Speed up se puede calcular como:

$$S_{UP} = \frac{CPI_V}{CPI_N}$$

$$CPI_V = 0{,}81 \times CPI_{ENT} + 0{,}19 \times CPI_{PFV} = 0{,}81 \times CPI_{ENT} + 1{,}9 \times CPI_{ENT} = 2{,}71 \times CPI_{ENT}$$

$$CPI_N = 0{,}81 \times CPI_{ENT} + 0{,}19 \times CPI_{PFN} = 0{,}81 \times CPI_{ENT} + 0{,}95 \times CPI_{ENT} = 1{,}76 \times CPI_{ENT}$$

$$\Rightarrow S_{UP} = 1{,}54$$

4) Suponer que se hicieron las siguientes mediciones para la ejecución de un determinado benchmark:
Frequencia de todas las operaciones de FP = 45 %
CPI promedio de la operaciones de FP = 4.85
CPI promedio de otras instrucciones = 1.73
Frequencia de FPSQR= 5 %
CPI de FPSQR = 20
Suponer que se intenta llevar el CPI de las operaciones FPSQR a 3.
a) Calcular el porcentaje de tiempo en la máquina no mejorada en el que el programa trabaja en operaciones de FP.
b) Calcular el porcentaje de tiempo en la máquina no mejorada en el que el programa trabaja en operaciones de FPSQR.
c) Calcular el porcentaje de tiempo en la máquina mejorada en el que el programa trabaja en operaciones de FP.
d) Calcular el porcentaje de tiempo en la máquina mejorada en el que el programa

trabaja en operaciones de FPSQR.
e) Calcular en cuanto se aceleran las operaciones de FPSQR.
f) Calcular la aceleración global obtenida.

Nota: FP: Punto Flotante, FPSQR: Raíz Cuadrada en Punto Flotante.

a) Calculamos el CPI promedio del programa antes de la mejora:

$$\overline{CPI} = 4{,}85 \times 0{,}45 + 1{,}73 \times 0{,}55 = 3{,}134$$

Por lo tanto, el porcentaje de tiempo que estamos trabajando con operaciones de punto flotante será:

$$\%FP = \frac{4{,}85 \times 0{,}45}{3{,}134} \times 100 = 69{,}3\,\%$$

b) Aplicando el mismo criterio que antes:

$$\%FPSQRT = \frac{20 \times 0{,}05}{3{,}134} \times 100 = 31{,}9\,\%$$

c) Necesitamos conseguir el nuevo CPI promedio, para esto, necesitamos el nuevo CPI promedio de punto flotante, y para calcularlo, es necesario obtener el CPI de las instrucciones de punto flotante que no son la raiz cuadrada. Cabe destacar que, ya que las operaciones de FPSQRT son el 5 % de las totales, y el 45 % son las de FP, entonces las operaciones de FPSQRT serán el 11.1 % de las operaciones de FP (si en 100 operaciones tengo 45 de FP y 5 de ellas son de FPSQRT, en 100 operaciones de FP tengo 11.1 operaciones de FPSQRT). Con esto puedo hacer el cálculo necesario:

$$CPI_{FPV} = 4{,}85 = 0{,}889 \times CPI_{otros-FP} + 0{,}111 \times 20 \rightarrow CPI_{otros-FP} = 2{,}96$$

Con la mejora:

$$CPI_{FPN} = 0{,}889 \times 2{,}96 + 0{,}111 \times 3 = 2{,}97$$

Calculamos el CPI promedio:

$$\overline{CPI_N} = 2{,}97 \times 0{,}45 + 1{,}73 \times 0{,}55 = 2{,}288$$

Entonces el porcentaje de tiempo será:

$$\%FP = \frac{2{,}97 \times 0{,}45}{2{,}288} \times 100 = 58{,}4\,\%$$

d) Usando el mismo criterio de antes:

$$\%FPSQRT = \frac{3 \times 0{,}05}{2{,}288} \times 100 = 6{,}56\,\%$$

e) Este punto se hace directamente con los datos del enunciado:

$$Sup_{FPSQRT} = \frac{T_V}{T_N}$$

Como el IC no cambia, ni tampoco el clock:

$$Sup_{FPSQRT} = \frac{CPI_{FPSQRT-V}}{CPI_{FPSQRT-N}} = \frac{20}{3} = 6{,}67$$

f) Dado que el IC y el clock no cambian:

$$Sup_{Total} = \frac{CPI_V}{CPI_N} = \frac{3{,}134}{2{,}288} = 1{,}37$$

4. Caché y Memoria Virtual

1) Dada una CPU MIPS con hardware de manejo memoria virtual se pide:
a) Calcular el tamaño del espacio de direcciones virtuales.
b) Dar la cota máxima de la cantidad de memoria que efectivamente puede usar un programa.
c) Calcular el tamaño en bytes de la tabla de traducción de páginas.
d) Calcular el tiempo de acceso promedio a memoria para una cache direccionada por direcciones virtuales.
Datos:
RAM = 512 MB
SWAP = 256 MB
5 bytes por entrada en tabla de traducción de página.
Tasa de desaciertos de la memoria cache = 5 %
Tiempo de acierto = 2 ns.
Tiempo de copiar un bloque completo de memoria principal a memoria cache = 40ns
Tasa de desaciertos del TLB = 3 %
Tiempo de actualizar una entrada en TLB = 15 ns.
Tasa de falla de páginas = 0.0001 %
Penalidad por falla de página = 3 mseg.
Aclare cualquier hipótesis que realice, y justifique las respuestas.

a) Asumiendo MIPS32 digo: 2^{32} posiciones, por lo tanto: 4 GigaBytes.

b) Justamente, el tamaño de direcciones virtuales es lo máximo que puede utilizar un programa (quitando despreciables bytes de dónde esté ejecutando el programa).

c) No sé cómo pretende que haga esto sin tener conocimiento del tamaño de cada página virtual. Si el tamaño fuera 2^b, entonces la cantidad de entradas en table sería de 2^{32-b}, y para el tamaño que ocupa en memoria, se lo multiplica por los 5 bytes que ocupa cada entrada en la tabla .

d) Resolvemos separando un poco la cuestión, sabiendo que el tiempo de TLB se toma en cuenta solo si hay Miss:

$$T_{acc} = T_{Cache} = T_{Hit-Cache} + MR_{Cache} MP_{Cache}$$

Resolvemos cada uno por separado:

$$MP_{Cache} = 40ns + T_{TLB}$$

$$= 40ns + T_{Hit-TLB} + \%miss_{TLB} \cdot MP_{TLB} = 40ns + 0{,}03\,(15ns + 0{,}000001 \times 3000000ns)$$

Dado que no nos dicen el tiempo de Hit de la TLB, asumo que es 0:

$$\Rightarrow T_{TLB} = 40ns + 0{,}54ns = 40{,}54ns$$

$$\Rightarrow T_{Cache} = T_{Hit-C} + \%miss_C \cdot MP_C = 2ns + 0{,}05 \times 40,54ns = 4,027ns$$

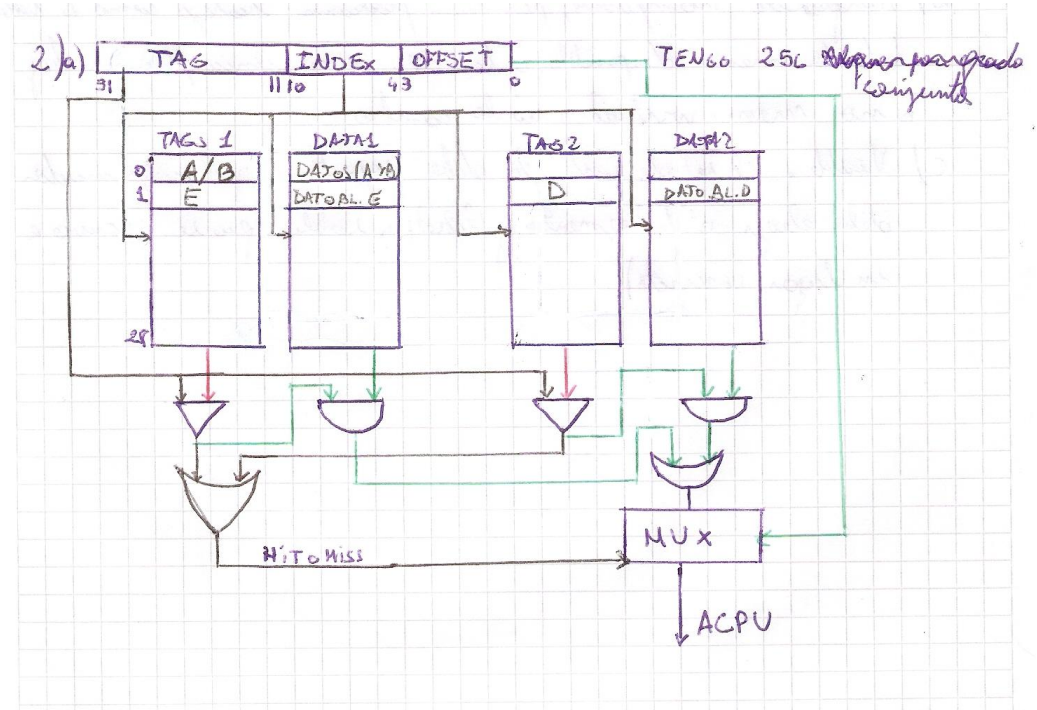
$$\Rightarrow T_{acc} = 4,027ns$$

2) Se tiene una memoria cache de 8Kbytes, asociativa por conjuntos de grado 2, bloque de 16 bytes y política de reemplazo LRU.
a) Realice un diagrama en bloques detallado de la memoria cache (incluir el área de tags).
b) Dar las direcciones para cinco datos A, B, C, D, E de manera tal que accedidos en el orden dado por una CPU MIPS se cumpla con lo siguiente:

1. **El acceso a B es hit.**
2. **C mapea en un conjunto distinto al correspondiente a A.**
3. **El acceso a E es miss.**

c) Indicar en el diagrama del punto 'a' el contenido final de la memoria cache (área de datos y área de tags).
Las direcciones se deben dar en hexadecimal. No se pueden hacer hipótesis acerca del contenido previo de la cache. Justificar sus respuestas.

a) Resuelvo ya indicando el resultado en c) (ver punto b):



b) Para que B sea hit, no pudiendo asumir que ya estaba su bloque en caché, es necesario que el bloque en el que está la posición B sea el mismo en el que está la posición A, así, aunque para este sea un miss (o no), para B necesariamente será hit. Además, para que E sea miss indpendientemente de los valores iniciales de la caché, es necesario que mapee al mismo conjunto que D, que bien puede ser al mismo que A/B o al de C (elijo por este último). Sabiendo que tengo 256 conjuntos (7 bits de índice) y cada bloque ocupa 16 bytes (4 bits de offset), elijo:

$$\text{A} = 0\text{x}0800 \ \text{B} = 0\text{x}0804 \ \text{C} = 0\text{x}0810 \ \text{D} = 0\text{x}1810 \ \text{E} = 0\text{x}2810$$

3) Colocar en los casilleros de la siguiente tabla la cantidad de bytes accedidos para cada uno de los subsistemas correspondientes a cada columna de la mencionada tabla, durante el proceso de acceder a un dato en memoria solicitado por la CPU. Justificar su respuesta con un breve párrafo para cada fila. CPU MIPS R2000, El acceso es una escritura en memoria de un dato de 4 bytes. L1 es direccionado por direcciones virtuales, es write through con bloque de 64 bytes. L2 es wirte back con bloque de 64 bytes. Los marcos de página son de 4 Kbytes. Aclare sus hipótesis.

Evento	Cache L1	Cache L2	Memoria Principal	TLB	Tabla de traducción de páginas	Área de swap
Hit L1						
Miss L1 y Hit L2						
Falla de Página						

Primero relleno la Tabla, luego explico, para esto asumo que la L2 está físicamente direccionada y asumo que cuando se consulta a la TLB es hit (pues no tengo información acerca de las PTE, porque no conosco siquiera la cantidad de memoria física), excepto en la Page Fault:

Evento	Cache L1	Cache L2	Memoria Principal	TLB	Tabla de traducción de páginas	Área de swap
Hit L1	64B	64B	0	?	0	0
Miss L1 y Hit L2	64B	64B	0	?	0	0
Falla de Página	64B	64B	64B	?	¿?	4KB

Explicación: En donde pongo signos de pregunta '?', es porque el dato en TLB es una PPN, que depende de la memoria física de la computadora, de la cual no tengo datos. En todos los casos vale lo mismo (2^{cant}). En el otro '¿?' vale lo que sea el largo de la PTE.
Primera Fila: En primer lugar, siempre estamos buscando un dato en la caché L1, entonces será necesario acceder a todo el bloque (64B), mínimamente para corroborar si el elemento está (hit). Como la caché está virtualmente direccionada, no es necesario ir a la TLB a hacer la traducción. Dado que se trata de una escritura en una Write-Through, se escribirá al siguiente nivel (L2), pero como esta es una Write-Back, no le irá a escribir a memoria, a menos que el bloque desplazado haya estado dirty (asumo que no). Es necesario acceder a la TLB para traducir la VPN de la dirección.
Segunda Fila: El caso es el mismo al anterior, pues se trata de una escritura, y más aun porque en este caso hay HIT en L2.
Tercera Fila: En este caso, como hubo una page fault, no puedo asumir justamente que no hay conexión con la memoria. En este caso, el dato reemplazado en cache estaba dirty, haciendo que se tenga que sobreescribir un dato de memoria principal que justamente no se encuetra disponible, incurriendo en un fallo de página. Entonces, es claro que hay un miss en la TLB, y que por lo tanto, al tratar de levantar la página a memoria, se intercambian los 4KB correspondientes.

Este ejercicio tiene toda la pinta de que era una lectura en vez de una escritura (en cuestión de la tabla, porque los misses te los pasas por los huevos).

4) Indicar para cada uno de los siguientes componentes del sistema de memoria, si el mismo es transparente o no a la arquitectura de programación. Justifique sus respuestas.
a) Memoria cache de nivel 1.
b) Memoria virtual paginada.
c) Memoria virtual segmentada.

a) Transparente. Justamente la idea de la Cache es serlo. La CPU accede a cache como si fuera memoria principal, no distingue entre una y otra, por lo que no ve la diferencia.
b) Transparente. Pues quien programa trabaja como si tuviera toda la memoria disponible posible (o sea, la memoria virtual), y como si la CPU corriera un único proceso. Si hay o no implementada memoria virtual, no le cambia, mientras no haya problemas luego.
c) Visible. El set de instrucciones lo debe soportar especialmente, y el programador debe elegir un número de segmento y el offset adecuado en cada oportunidad.

5) Se tiene una memoria cache de 4Kbytes, asociativa por conjuntos de grado 4, bloque de 16 bytes y política de reemplazo LRU.
a) Realice un diagrama en bloques detallado de la memoria cache (incluir el área de tags).
b) Dar las direcciones para cinco datos A, B, C, D, E, F de manera tal que accedidos en el orden dado por una CPU MIPS se cumpla con lo siguiente:

1. **A mapea en el conjunto 1**
2. **B produce hit**
3. **C no pertenece al bloque que contiene a A y mapea en el mismo conjunto en el que mapeo A.**
4. **El acceso a F es miss**

c- Indicar en el diagrama del punto 'a' el contenido final de la memoria cache (área de datos y área de tags).
Las direcciones se deben dar en hexadecimal. No se pueden hacer hipótesis acerca del contenido previo de la cache. Justificar sus respuestas.

a- idem 2a.

b- Los bits de offset son los últimos 4 bits (1 dígito Hexadecimal). Los bits de index dependen de la cantidad de conjuntos = 4Kbytes/ (4*16 bytes) = 64 conjuntos, es decir, 6 bits de Index.
Dada condición 2, B debe estar en el mismo bloque que a.
Entonces decimos para que, por condición 1, A mapee en conjunto 1: A=0x400; B=0x404. A tiene un 1 en el último bit de Index.
Dada la condición 3, los 6 bits de Index deben coincidir con los de A (todos ceros).
Entones: C=0x1400.
Luego, para cumplir la condición 4, debemos hacer que C y D mapeen al conjunto 1, desalojando (posiblemente) el dato F, que también debe mapear al conjunto 1.

Entonces: D=0x2400; E=0x3400; F=0x4400.

6) Indicar la cantidad de bytes de información que se transfieren entre los distintos pares de dispositivos dados en la siguiente tabla, para los eventos indicados y la configuración del sistema de memoria dada:
Configuración: Procesador MIPS. Cache L1 write through, cache L2 write back, tamaño de bloque L1 32 bytes, tamaño de bloque L2 32 bytes, se debe supponer que los bloques de la cache L2 que entren en juego están modificados.

	CPU-Cache L1	Cache L1 - Cache L2	Cache L2 - Memoria
Acceso a dato de instrucción lw . Miss en L1 y en L2	4B	32B	64B
Acceso a dato de instrucción sw. Miss en L1 y Hit en L2	4B	32B	0
Búsqueda de instrucción addi. Hit en L1 y en L2	4B	0	0

Siempre entre CPU-L1 va a ser de 4B (ya sea de escritura o de lectura) porque la CPU accede como si fuera memoria principal. Esto es independiente del tamaño del bloque. Entre Ambas Caché serán 32B pues en ambos casos vale lo mismo el tamaño del bloque (si fuera distinto habría que diferenciar caso de lectura o escritura, pues en la escritura el L1 le pasa su bloque (en caso de necesitarlo) a la L2, y en caso de lectura, el L2 le pasa el bloque al L1. En el caso entre L2 y memoria se transmite el doble del bloque de L2 pues hay una lectura (se obtendrán 32B de datos de lectura) y además, como los datos dentro de la L2 estan modificados (dirty) será necesario enviar 32B (un bloque) a memoria, por lo tanto acá hay un ida y vuelta.

7) Se tiene una TLB de 8 entradas con política de reemplazo LRU. Una aplicación realiza la siguiente secuencia de referencias a páginas virtuales contiguas: P0,P1,P2,P3,P4,P5,P6,P7,P8,P9. Dicha secuencia se repite en forma cíclica por un período muy largo de tiempo.
a- Calcular la tasa de desaciertos para el caso en que la TLB es de mapeo directo.
b- Ídem punto a- pero considerando que es asociativa de grado 2.
c- Ídem punto a- pero considerando que es totalmente asociativa.
Justificar sus respuestas.

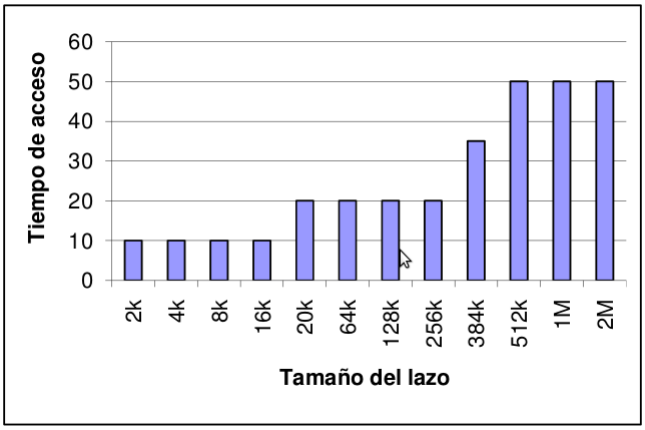
a) Haciendo un par de dibujos podemos ver que por cada ciclo, como tenemos 10 páginas y sólo 8 entradas, tendremos 4 misses por ciclo, entonces es un total del 40 %.

b) Idem anterior, con un par de dibujos podemos notar que tendremos 6 misses por ciclo, por lo tanto se tiene un 60 %.

c) Idem anterior, todavía más sencillo, con tener un único elemento más que el que soporta la cache es suficiente para que se produzca trashing total, por lo tanto: 100 %. (El elemento 0 se ve desplazado por el 8, el 1 por el 9, el 2 por el 0, el 3 por el 1, etc....)

8) En el siguiente gráfico se dan los tiempos de acceso a memoria promedio obtenidos cuando una CPU ejecuta el modelo del lazo para lazos de distinto tamaño

en una computadora bajo test. Calcular justificando claramente:
a) la cantidad de niveles de memoria cache existentes y el tamaño de cada nivel.
b) El grado de asociatividad de cada nivel.



La idea sería esta: viendo que hasta los 16K se mantiene un acceso constante, y sabiendo que en un lazo, se genera trashing en una FA con un único elemento más que el tamaño de la caché, podemos decir que hay una FA de 16KB como nivel 1 (grado 1). El salto que se produce (y lo constante que es) hasta los 256KB es porque allí hay otra Caché, y como siempre tenemos que pasar por la primera caché, esta dará miss, y luego tenemos que acceder a la segunda. Ahora, no podemos volver a decir que tiene grado 1 de asociatividad porque no se explicaría lo que pasa en 384KB. Como ya se sabe de la práctica, si tuvieramos una de grado 2, el trashing se logra al tener 1.5 elementos respecto del tamaño de la caché, es decir, a 256Kb * 1.5 = 384Kb, con lo cual habría trashing completo, esta no puede ser la respuesta. Debemos estar hablado de una caché de Mapeo Directo, que lo alcanza al tener dos veces la cantidad de elementos. Entonces, el miss rate no es del 100 %, y por eso no llega hasta el tope de tiempo, que trata del tiempo de acceso a memoria principal. ¿Por qué no una de grado 4? Porque en una de grado 4, luego de hacer un dibujo, se consigue trashing completo teniendo un 25 % más de elementos nada más, por lo que el de 384KB necesariamente estaría al tope de tardanza.

9) Calcule la cantidad de desaciertos para cada uno de los tipos dados por el modelo de las "3 C"para la siguiente secuencia de referencias a memoria producida por un microprocesador MIPS:

```
1.  0x00000000
2.  0x00000004
3.  0x00000008
4.  0xf0000000
5.  0xf0000010
6.  0xf0000020
7.  0xf000001c
8.  0xfff00000
9.  0x0000000c
10. 0x00000100
```

Datos de la memoria cache: asociativa por conjuntos de dos vías, 4 bloques de 16 bytes. Se debe suponer que ninguna de las referencias está contenida en la cache

al momento del comienzo de la secuencia.

Haciendo los dibujos correspondientes queda que:
1. Es miss compulsivo, mapea en conjunto 0.
2. Es Hit (con el bloque de 1).
3. Es Hit (misma razón).
4. Es miss compulsivo, mapea al conjunto 0.
5. Es miss compulsivo, mapea al conjunto 1.
6. Es miss compulsive, mapea al conjunto 0, desplazando al 1/2/3.
7. Es Hit (con el bloque de 5)
8. Es miss compulsivo, mapea en el conjunto 0, desplazando al bloque de 4.
9. Es miss de capacidad*, mapea en el conjunto 0 desplazando al bloque de 6.
10. Es miss compulsivo, mapea al conjunto 1.

*Es miss de capacidad, pues este miss se produciría también en una FA de mismo tamaño que nuestra Caché (ver la cantidad de elementos que ya fuimos ingresando, teniendo solo espacio para 4 bloques en total).

10) Modelos de las "3 C" para los desaciertos en memoria Caché.

Mucho gusto.. bueno, asumiendo que pedía que se explique, la idea es poder clasificar los misses en tres tipos:

- Compulsivos/Obligatorios: aquellos que se hacen por acceder a un bloque por primera vez. Dicho de manera "linda", serían los misses que se producen en una FA de tamaño infinito.
- Capacidad: aquellos que se hacen porque falte capacidad de la caché (o sea, se solucionarían si tuviéramos más espacio). Dicho de manera "linda", serían los misses que se producen en una FA de mismo tamaño a nuestra caché.
- Conflicto: el resto de los misses.

Dado que en algunos casos puede que los de capacidad excedan el porcentaje de miss real (pues con un sólo elemento de más ya tenemos trashing), se toman dos soluciones posibles: 1) ponemos un valor negativo de porcentaje en el de conflicto, para ajustar 2) Aparece el modelo "D3C", que dice básicamente lo mismo, pero con un pequeño cambio:

- Compulsivos/Obligatorios: los misses que se producen en una FA de tamaño infinito y en en la cache de estudio.
- Capacidad: los misses que se producen en una FA de mismo tamaño a nuestra caché y en la cache de estudio.
- Conflicto: el resto de los misses.

La primera técnicamente no cambia mucho, la verdad, pero la segunda si. Con eso sólo tomamos en cuenta los misses que aparecen realmente en nuestra caché, que es lo que importa.

11) Diferencias y similitudes entre memoria virtual paginada y memoria virtual segmentada. ¿Son visibles a la arquitectura del conjunto del instrucciones?, ¿por

qué?

Similitud: Ambos permiten simular memoria virtual, por ende, que un proceso dado cuenta con toda la memoria disponible para la arquitectura del set de instrucciones.

Diferencias:

- Para la paginada se necesita una sola palabra para acceder a memoria, para la segmentada se necesitan dos (segment + offset).
- La paginada es invisible al programador (pues se accede a todas las posiciones de igual manera), la segmentada no, pues el programador debe saber el número de segmento más el offset, teniendo cuidado de su uso.
- Reemplazar un bloque en la paginada es trivial, por ser todos del mismo tamaño, en el caso de la segmentada no, porque se necesita encontrar una porción de memoria de mismo tamaño en memoria para reemplazar.
- Hay porciones enteras de memoria que no se usan en la segmentada, siendo la separación visible externamente, mientras que para la paginada, la fragmentación es interna totalmente.
- La paginada esta pensada para un eficiente transporte de datos entre disco y memoria, mientras que en la segmentada es todo muy variable.

12) Estudie el comportamiento de un patrón de referencias a memoria que sigue el modelo del lazo. Clasifique los desaciertos usando el modelo de las "3C". Interprete los resultados obtenidos.
Datos: Lazo de 11 bloques. Cache de 8 bloques, 2 vías y reemplazo LRU.

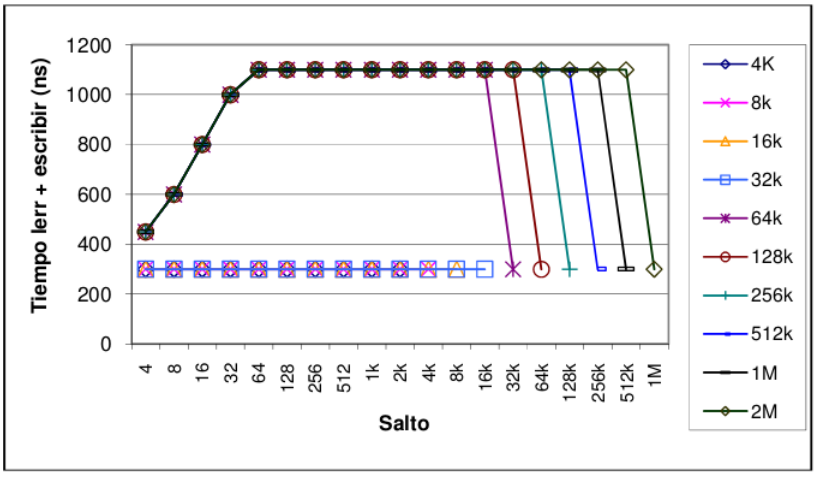
Haciendo unos breves dibujos queda que tenemos 9 misses en 11 accesos por ciclo, llegando a un 81.8 % de miss. Siguiendo el modelo de las "3C" tendremos un 100 % de capacidad, y -18.2 % de conflicto, mientras que en el modelo "D3C" tendremos 81.8 % de misses de capacidad y 0 de conflicto.
En todo esto estoy despreciando los 11 misses obligatorios, porque el bucle se ejecuta suficientes veces como para que sean despreciables.

13) Indicar para cada uno de los siguientes componentes del sistema de memoria, si el mismo es transparente o no a la arquitectura de programación. Justifique sus respuestas.
a) Memoria cache de nivel 1.
b) Memoria virtual paginada.
c) Memoria virtual segmentada.

Mismo que el ejercicio 4.

15) En una computadora se corre un test que referencia una zona contigua de memoria en forma de lazo utilizando un determinado salto entre las direcciones

sucesivas. Se ejecutan lazos de distintos tamaños con saltos de distintos tamaños y se mide el tiempo promedio que llevó hacer una lectura más una escritura. Los resultados obtenidos se muestran gráficamente a continuación, donde se usan distintos marcadores para lazos de distinto tamaño. Responder justificando los siguientes puntos:
a) cuantos niveles de memoria cache hay?
b) tamaño de cada nivel.
c) tamaño del bloque.
d) grado de asociatividad de cada nivel.



Bueno, este problema es saladito. Primero vamos por lo más *fácil*: vemos los lazos que están planchados en el piso entre 200 y 300 ns. De ahí viendo que el último que se mantiene constante es el lazo de 32KB podemos definir que la capacidad de un primer nivel de cache será exactamente 32Kb (no depende de los saltos, pues eso varía en tema de dónde caen los elementos, no cuantos hay). Digo primer nivel pues son los lazos más pequeños, por lo que necesariamente ese sería un primer nivel.
Luego, analizamos la subida que va desde los saltos de 4B hasta los 64B. Con esto podemos obtener el tamaño del bloque que decimos es justamente 64B. ¿Por qué decimos esto? porque viendo que más adelante la cosa se estaciona (en lo que sería la velocidad de una supuesta L2 o memoria principal), tendríamos que pensar cuál es la razón por la que a saltos pequeños no pasa. Esto sucede porque el bloque es lo suficientemente grande como para que, en primera instancia haya un miss, pero contenga varios de los siguientes elementos, entonces habrán algunos hits y miss que irán aumentando en cuanto vaya aumentando el salto (pues iremos yendo de bloque en bloque con mayor frecuencia, en vez de permanecer más *tiempo* en uno mismo). Cuando llegamos a 64B ya tenemos 100 % de miss rate, que tiene sentido que sea con un salto igual o mayor al tamaño del bloque. Por lo tanto, ya tenemos ese dato también. Ahora nos falta analizar por qué caen repentinamente los gráficos de los saltos más grandes. Esto se debe al nivel de asociatividad. En particular, se ve que la caída se produce en exactamente la mitad del tamaño del bucle, esto quiere decir que sólo se tendrán dos accesos. Por lo tanto, podemos decir que se trata de una asociativa de dos vías y listo. ¿Por qué no de cuatro? porque sino, podría también bancar hasta un cuarto del tamaño del hilo, cosa que no pasa, y con una de dos vías justamente esto dará miss. Por lo tanto, respondo:
a) Hay un único nivel de caché (o de L2 no podemos conseguir datos, excepto que todo será hit).
b) L1 será de 32KB de capacidad.

c) El bloque será de 64B.
d) El grado de asociatividad es 2.

16) Para un sistema que utiliza el esquema de memoria virtual paginada con marcos de página de 4 Kbytes, diseñar una memoria cache de 32 Kbytes de tamaño y bloques de 16 bytes y que sea indexada virtualmente y tageada físicamente. Realizar un diagrama en bloques detallado del diseño realizado.

Nada, es hacer el méndigo dibujo, haciéndo énfasis en que el tag irá a traducirse mientras buscamos en el índice para luego comparar. Como no especifica el tipo de caché, elegiría una de direct map, para hacer la cosa más sencilla nada más.

17) a) Para un sistema que utiliza el esquema de memoria virtual paginada con marcos de página de 4 Kbytes, realizar un diagrama en bloques detallado de una TLB de 8 entradas totalmente asociativa con reemplazo LRU.
b) Dar los contenidos de todos los campos de la TLB del punto anterior, inmediatamente después de la ejecución de un benchmark que referencia la memoria como lo hace el modelo del lazo simple para un lazo de 64 Kbytes. Aclare sus hipótesis.

a) Será hacer simplemente de una asociativa con 8 entradas y listo, no hay mucho más que eso.

b) Con 64K de salto, incurrimos en 16 páginas. Pero como es lazo simple, accedemos a posiciones contiguas de memoria, haciendo que hayan $\frac{4KB}{4B} - 1$ hits, 16 veces, y 16 misses. El contenido final será justamente el de las últimas páginas, por ser totalmente asociativa (quedarán las páginas 8, 9... 15).

18) Explique cuáles son las ventajas de un sistema con memoria cache indexada virtualmente y taggeada físicamente. Que limitaciones pone este diseño en el tamaño de la cache.

Con un index virtual se puede ir buscando el índice correspondiente mientras se va traduciendo el tag correspondiente, para luego poder hacer la comprarción rápidamente. Además, en caso de necesitar ir a memoria a buscar el dato (en caso de miss), ya tenemos traducida la dirección (siendo el index y offset de la cache el offset de la página).
Asimismo, esto atrae una limitación en la capacidad de la caché: justamente necesitamos que tenga el index + offset igual al necesario para cubrir el offset de la página (o sea, el tamaño de la página). Por lo tanto, ya tenemos seteado este parámetro, y estamos limitados a cuánto de offset o index podemos llegar a poner.

19) De un ejemplo de de código MIPS para cada uno de los siguientes casos. Cada ejemplo debe ilustrar el principio de localidad para las referencias a memoria:
a-Localidad espacial de instrucciones.
b-Localidad espacial de datos.
c-Localidad temporal de instrucciones
d-Localidad temporal de datos.

a) Localidad espacial de instrucciones:

```
add r1, r2, r3
sub r2, r5, r4
add r4, r4, r1
```

b) Localidad espacial de datos:

```
lw r1, 0(r2)
lw r3, 4(r2)
add r4, r1, r3
sw 8(r2), r4
```

c) Localidad temporal de instrucciones:

```
loop:
add r1, r2, r3
sub r2, r5, r4
bne r2, loop
```

d) Localidad temporal de datos:

```
loop:
lw r1, 0(r4)
lw r2, 4(r4)
addu r3, r1, r3
sw 8(r4), r3
addu r1, r2, r1
sw  0(r4), r1
subiu r5, r5, 1
bne r5, loop
```

20)a) Definir conceptualmente los tipos de misses dados por la clasificación de las "3C".
b) Calcule los porcentajes para cada tipo de miss para una cache asociativa por conjuntos de 2 vias de 32 Kbytes de nivel 1 en base a los misses obtenidos y dados en la siguiente tabla:

Cache	Totalmente Asoc.	8 vías	4 vías	2 vías	Mapeo directo
32KB	3 %	5 %	6 %	8 %	15 %
1MB	0.2 %	0.3 %	0.5 %	2 %	3 %

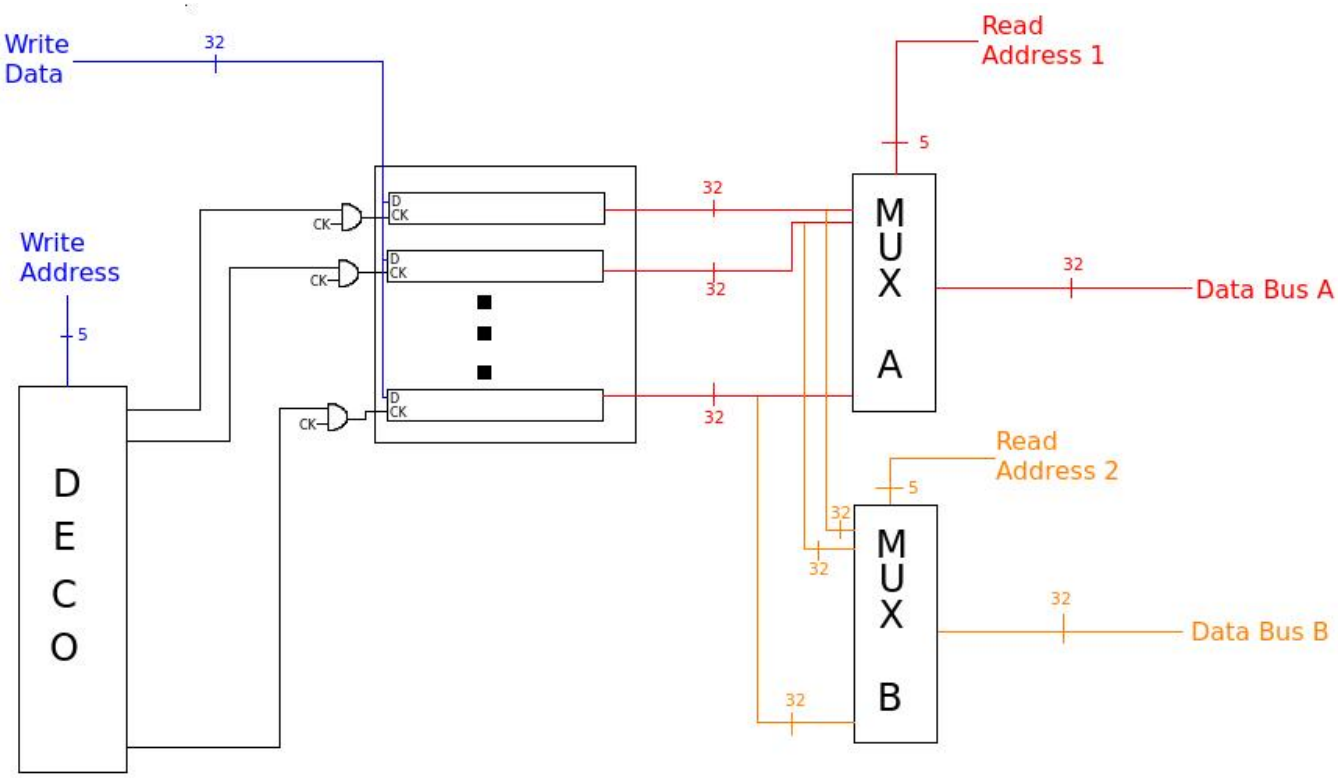
Bueno, acá nos dan mucha cosa al pedo. Lo único que nos importa, es que como una cache de 1MB es muy grande (podemos considerarla *infinita*). Con esto tenemos los misses obligatorios (0.2 %), por otro lado, tenemos el dato del porcentaje de miss en caso de una FA de mismo tamaño a la nuestra (3 %), y el total de los misses de nuestra caché (8 %), por lo tanto:

- Misses Compulsivos: 0.2 %

- Misses de Capacidad: 3 %
- Misses de Conflicto: 4.8 %
- Misses totales: 8 %

5. Pipeline

1) Realice un diagrama en bloques detallado de un Register File adecuado para una CPU MIPS con una implementación pipeline de 5 etapas (IF, ID, EX, MEM, WB). Justificar.



La idea es que la señal de rs va al MUXA, a quien le entran los $32 \times 32 reg$ cables provienientes de los registros. Esa señal elige la salida especifica. Lo mismo sucede con rt con MUXB. Para la escritura, hay un MUXC que le envia una señal de un bit a cada registro para indicar cual es el de escritura (a partir de la señal de 5 bits), mientras que de afuera viene lo que hay que escribir (solo se escribirá en el registro indicado por el decodificador).

2) Describir cómo funciona un esquema predicción de saltos para el pipeline MIPS de 5 etapas, asumiendo que la predicción de saltos dinámica se realiza en la etapa de IF. Suponiendo que los saltos se resuelven en la etapa de EX del pipeline conteste las preguntas siguientes y proporcione las explicaciones para cada una. Proporcione cualquier diagrama que usted piense será provechoso para explicar su respuesta.
a. ¿Cuántos ciclos se pierden si un branch se predice correctamente no tomado?
b. ¿Cuántos ciclos se pierden si un branch se predice correctamente tomado?
c. ¿Cuántas ciclos se pierden si un branch se predice incorrectamente no tomado?
d. ¿Cuántas ciclos se pierden si un branch se predice incorrectamente tomado?

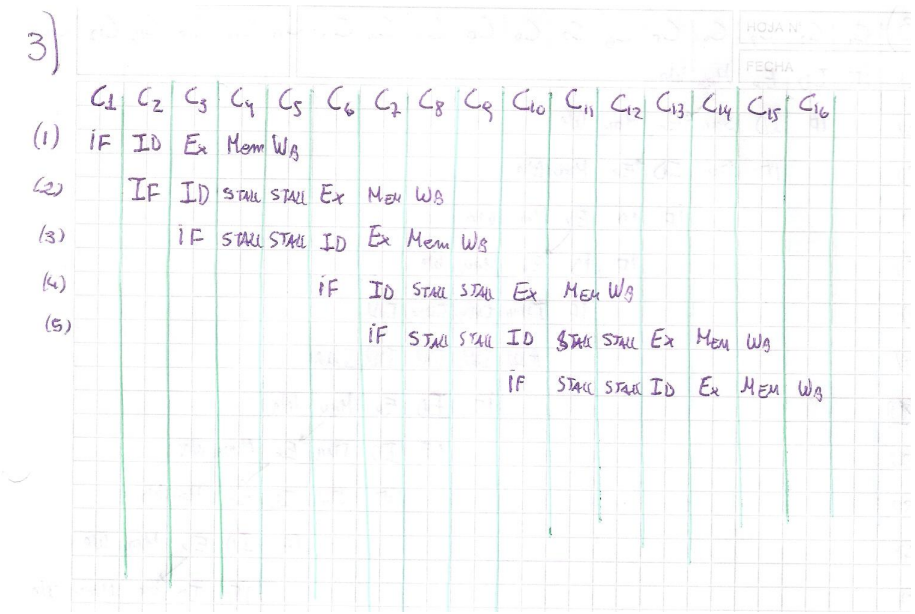
En la etapa IF se obtiene a partir de los bits de la instrucción la dirección a dónde saltará. Además, con el PC, antes de ir a memoria de Instrucciones ya se sabe que predicción se hará.

Por lo tanto, a partir de ya tener la dirección de la siguiente posición a ejecutar (o mejor dicho, el offset al PC), se indica que será esa la siguiente posicion a ejecutar (cambiando el PC) o el PC+4, dependiendo del valor de la predicción. Si fue correctamente predicho, se termina la cuestión luego de la etapa EX. Sino, luego de la etapa EX será necesario hacer FLUSH del pipeline para no ejecutar instrucciones incorrectas, perdiendo así dos ciclos.
a) 0
b) 0
c) 2
d) 2

3- Dibujar un diagrama de ejecución pipeline y calcular la cantidad de ciclos de reloj que insume la ejecución del siguiente fragmento de programa para los siguientes casos en un procesador MIPS que utiliza el esquema del salto demorado (1 delay slot) y un pipeline de 5 etapas:

```
ADD R1, R2, R3
LW R4, 33(R1)
ADDI R4, R0, 100
SLTI R5, R4, 500
BGTZ R5, FIN /* ESTE SALTO ES NO-TOMADO*/
SLL R6, R5, 2
```

Asumo que, como no hay mención de hardware de fowarding, asumo que no hay implementado, pero si que se maneja la escritura y lectura en el register file separandolos en distintos semiciclos.



Por lo tanto, asumiendo que el SLL fue puesto por el compilador para aprovechar el Delay Slot, este ciclo tarda 16 ciclos en terminarse.

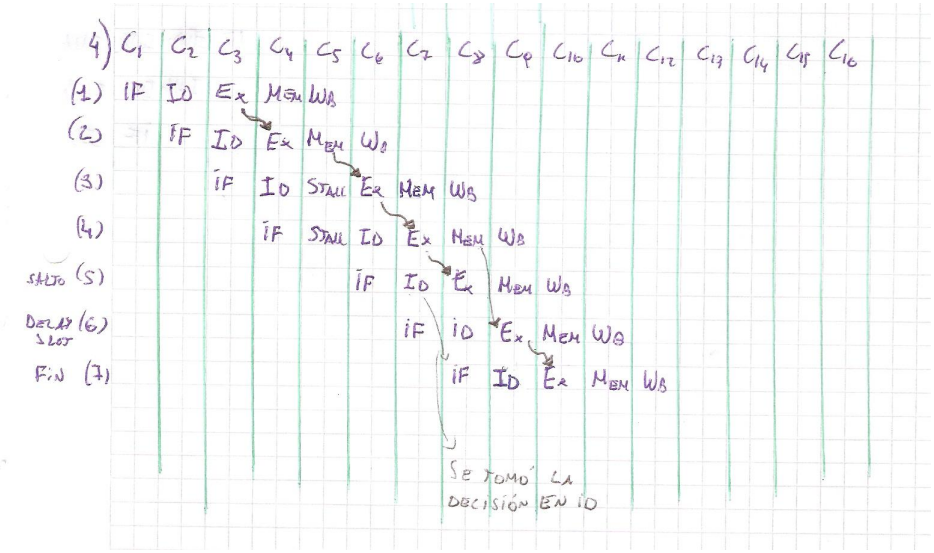
4) Dibujar un diagrama de ejecución pipeline y calcular la cantidad de ciclos de reloj que insume la ejecución del siguiente fragmento de programa para los

siguientes casos en un procesador MIPS que utiliza el esquema del salto demorado (1 delay slot) y un pipeline de 5 etapas:

```
ADD  R1, R2, R3
LW   R4, 33(R1)
ADDI R4, R4, 100
SLTI R5, R4, 500
BGTZ R5, FIN      /*Este salto es TOMADO*/
SLL  R6, R5, 2
...
FIN:
ADD R7,R6,R8
```

a) No se dispone de ningún tipo de hardware de bypassing o forwarding.
b) Se dispone de todo posible hardware de bypassing o forwarding.

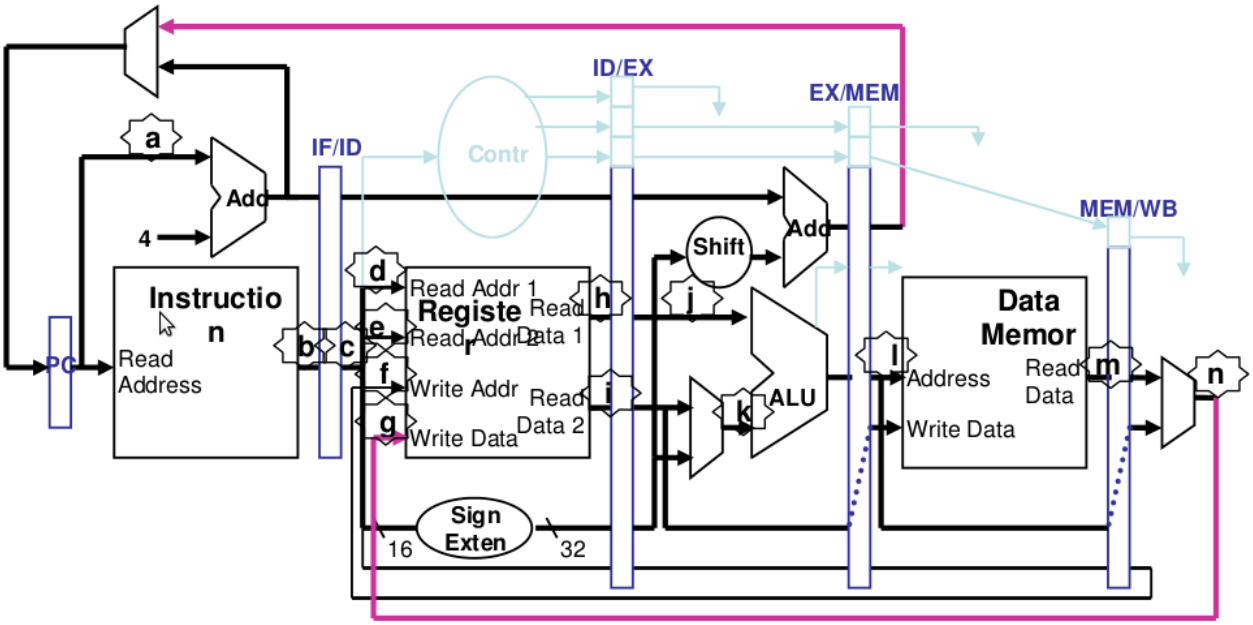
El a) sería muy similar al del ejercicio anterior, así que solamente hago el b)



5- El siguiente dibujo corresponde a una implementación pipeline para un procesador MIPS. a) Indique para cada letra indicada sobre una conexión, el valor binario establecido en dichas líneas durante el transcurso del quinto ciclo de reloj . Por simplicidad de lectura los valores binarios pedidos deben darse en hexadecimal.

```
0x400000 sw  $1,  24($2)
...      lw  $3,  36($4)
         add $5,  $6, $7
         add $8,  $9, $10
         add $11, $12, $13
```

Datos: $1 contiene 0x2000, $2 contiene 0x3000, $4 contiene 0x6000, $6 contiene 0x8000, $7 contiene 0x7700, $9 contiene 0x9900, $10 contiene 0x10100



Dado que estamos ya en el 5to ciclo, en a) irá el PC actual, por lo tanto, el que corresponde a la última instrucción. Por lo tanto, será 16 posiciones siguientes a la posición de la primera instrucción, por lo tanto, 0x400010. En B irá el binario de la instrucción 5 (add $11, $12, $13). De C a I estamos en la etapa ID, por lo que estamos trabajando con la 4ta operación. En C deberá ir el binario de la instrucción 4 (add $8, $9, $10). En D el binario del operador 1 (01001). En E el binario del operador 2 (01010). F indica el registro de escritura de la instrucción que se encuentra en la etapa de WB, o sea, el destino de la primera operación, en este caso, como no hay escritura en registro, se indicará el registro 0. En G irá el dato a guardarse en tal registro, también proveniente de la etapa WB, pero como no hay nada que escribir, será 0. En H irá el valor del registro 9 (0x9900) mientras que en I irá el valor del registro 10 (0x10100). Para J y K estamos en la etapa de EX, por lo que tenemos que tener en cuenta el camino de la tercera instrucción. En J irá el primer operando, o sea el contenido del registro 6 (0x8000), mientras que en K irá, o bien la constante con signo extendido (que en este caso no hay) o el contenido del segundo operando si lo hay (que en este caso, si). Entonces, en K está el valor del registro 7 (0x7700). En L y M estamos en la etapa de memoria. Para M tenemos el dato de la posición de memoria de donde deseamos leer (pues se trata de la segunda operación), o sea, 36 lugares más a la posición indicada por el registro 4 (0x6000 + 36 = 0x6024). En M irá el dato a leer, que será el contenido de aquella posición, cosa que desconocemos. En N estamos ya en la etapa de WB, por lo que irá el mismo valor que en G. En este caso, 0.

6) Explique como se resuelven los riesgos RAW,WAW y WAR en una implementación MIPS superescalar que utiliza el algoritmo de Tomasulo.

WAW y WAR: Se solucionan estos problemas utilizando un renombrado de registros. Esto quiere decir, vamos usando un cierto registro físico como si fuera el que queremos. Técnicamente, a nosotros no nos importa si el registro 10 es efectivamente el registro 10, sino que simplemente todo mantenga coeherencia (o sea, si yo pido que el registro 10 tenga un cierto valor, y se lo ponen al 15, mientras luego siga teniendo coeherencia, está todo bien). Lo que se hace, es tener

una tabla de registros libres para utilización. Si un registro no se puede usar, aunque me lo hayan pedido, yo doy otro a cambio, e indico que el nombre lógico de ese argumento es el que me pidieron (entonces, en vez de conectar eléctricamente en el Reg file directamente la señal proviente de la instrucción del rs y rt, se pasa por un *traductor* (Register Map) para saber cuál es el verdadero registro a utilizar). En cuánto se ve que un registro se deja de usar, se lo libera para que pueda ser usado por otro nombre lógico. A priori esto no tiene mucho sentido, pero en un procesador superescalar, o más aún en un multihilo, es muy importante tener este tipo de cuidados.

RAW: Se solucionan teniendo copiados los punteros a las estaciones de reserva que irán prouciendo los datos necesarios. De esta manera, se esperará a que estos datos estén listos (utilizando ciclos de stall mientras tanto), reemplazando luego el puntero con el valor real.

6) Riesgos de datos. Clasificación de los riesgos de datos. Análisis de todos los tipos de riesgos de datos e ilustración con ejemplos de secuencias de instrucciones para el procesador MIPS con implementación pipeline de 5 etapas (IF, ID, EX, MEM, WB).

Si, hay dos ejercicios 6...
Podrían haber 4 tipos de riesgos de datos: RAW, WAW, WAR y RAR. En particular, mientras no haya opearciones multiciclo o un superescalar metido en el medio, los únicos que joden son los riesgos RAW (Read after write). Un ejemplo donde se puede solucionar con fowarding:

```
add r3, r1, r2
add r5, r4, r3
```

Ahora, la siguiente operación se le puede mejorar los ciclos perdidos usando fowarding, pero es imposible no perder un ciclo con stall:

```
lw  r3, 0(r1)
add r5, r2, r3
```

Dado que se obtiene el dato de r3 recién de que la primera instrucción pasa el ciclo de memoria.

7)Riesgos estructurales. Definición. Describa dos ejemplos de riesgo estructural y la solución para eliminarlos.

Los riesgos estrucurales son aquellos problemas que pueden ocacionarsele al pipeline cuando instrucciones seguidas (seguidas significa que se encuentren dentro de la línea de montaje al mismo tiempo, no que venga una después de la otra), necesitan utilizar el mismo hardware. En algunos casos, se puede arreglar agregando puertos que permitan a un hardware ser utilizado más de una vez por ciclo (ej: el Reg. File, que en el primer semiciclo puede escribir, y en el segundo leer). Otra forma sería duplicando el hardware (Ej: tener ALU especializada, como la que suma al PC un 4, separada de la ALU principal). Cuando duplicar no es una opción, puede separarse el Hard dependiendo del momento de utilidad (Ej: tener la memoria principal separada entre memoria de instrucciones y memoria de datos). Como última alternativa, se puede utilizar un ciclo de stall para terminar de evitar estos riesgos (como puede pasar en algunos

casos al utilizar una arquitectura superescalar).

8- Escriba un fragmento de código MIPS que utiliza el esquema de salto demorado con un único delay slot que es ocupado por una instrucción NOP. El código debe ser tal que el delay slot puede ser llenado con una instrucción previa al branch, una del fall through y una del target. Rescriba el código original reemplazando la NOP con la instrucción correspondiente a cada uno de los tres casos pedidos.

```
loop:  addu r1, r2, r3
       addu r4, r4, r5
       subu r2, r2, r4
       addu r1, r1, r5
       bne  r2, loop
       nop
       addi  r1, r1, 100
```

Podemos transformarla utilizando un From-Before:

```
loop:  addu r1, r2, r3
       addu r4, r4, r5
       subu r2, r2, r5
       bne  r2, loop
       addu r1, r1, r5
       addi  r1, r1, 100
```

Por un Target:

```
loop:  addu r4, r4, r5
       subu r2, r2, r5
       addu r1, r3, r5
       bne  r2, loop
       addu r1, r2, r3
       addi  r1, r1, 100
```

Por un Fall-Through:

```
loop:  addu r1, r2, r3
       addu r4, r4, r5
       subu r2, r2, r5
       addu r1, r1, r5
       bne  r2, loop
       addi  r1, r1, 100
```

(Vemos que en este ultimo, el dato de r1 igual se sobreescribirá).

9) Se da a continuación un código para un procesador MIPS. El procesador que ejecutará este código se implementó con el esquema de salto demorado y requiere de un "delay slot". a) Evitar el riesgo de control utilizando instrucciones NOP. b) Aprovechar el delay slot utilizando instrucciones previas al branch. c) Aprovechar el delay slot utilizando instrucciones del fall through.

```
        ...
LABEL:  lw $5, 30($6)
        ...
        ...
        ...
        subiu $2, $2, $1
        addiu $3, $3, 1
        bne $4, $3, LABEL
        subiu $4, $4, 1
        addiu $5, $5, 12
```

a)

```
        ...
LABEL:  lw $5, 30($6)
        ...
        ...
        ...
        subiu $2, $2, $1
        addiu $3, $3, 1
        bne $4, $3, LABEL
        nop
        subiu $4, $4, 1
        addiu $5, $5, 12
```

b)

```
        ...
LABEL:  lw $5, 30($6)
        ...
        ...
        ...
        addiu $3, $3, 1
        bne $4, $3, LABEL
        subiu $2, $2, $1
        subiu $4, $4, 1
        addiu $5, $5, 12
```

c)

```
        ...
LABEL:  lw $5, 30($6)
        ...
        ...
        ...
        subiu $2, $2, $1
        addiu $3, $3, 1
        bne $4, $3, LABEL
        addiu $5, $5, 12
        subiu $4, $4, 1
```

10) Cuales son las características de la arquitectura del conjunto de instrucciones MIPS que facilitan la implementación pipeline.

Las características son:

- Todas las instrucciones tienen el mismo tamaño, por lo que los campos se encuentran siempre entre los mismos bits (EJ: el OPCODE siempre están entre el bit 31 y 26). Esto ayuda a sistematizar la obtención de datos de la instrucción.
- Se cuenta con muchos registros. Sin esto, abundarían los riesgos RAW.
- Solamente se accede a memoria de datos en el caso de loads y store, pudiendo dejar eso en una etapa específica. Si fuera una arquitectura memoria-memoria, habría que tener acceso a memoria para obtener los operandos, y habría que tener una memoria que permita que se lean varios elementos a la vez (y también poder escribirle a la vez, pues es un hardware que no se puede duplicar).
- Las instrucciones son simples (esa es la idea de las arquitecturas reg-reg), lo cual hace que sea sencillo poder subdividir la acción en pequeñas etapas (e inclusive, poder aplicar hiperpipeline, llegado el caso).

11) Idéntico al ejericio 5, con otros números.

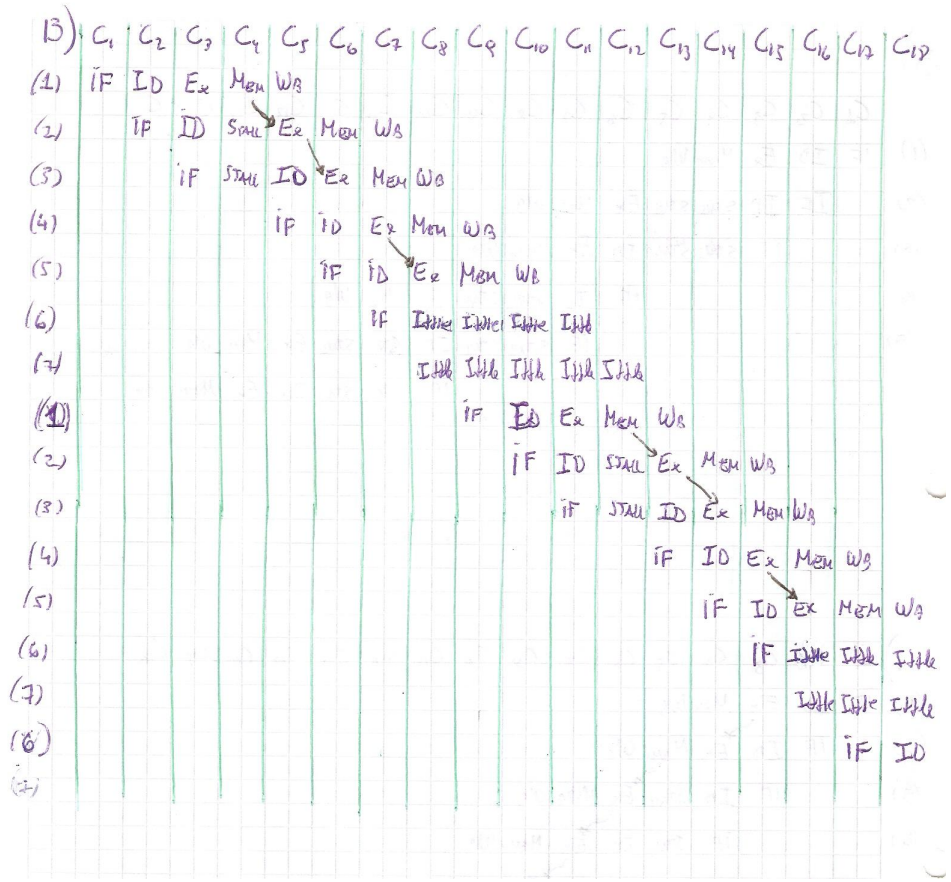
12) Dibujar el datapath completo para la arquitectura MIPS pipeline de 5 etapas. Incluir el forwarding ALU-ALU y Memoria-ALU

Esto es tan simple como mirar la carpeta.. no creo que nadie necesite más ayuda que eso.

13) Escriba un diagrama de tiempo pipeline (IF ID EX MEM WB) para un procesador MIPS que ejecuta dos iteraciones del siguiente código. Suponer todo el forwarding necesario, indicar con flechas en el diagrama el uso de este forwarding cuando corresponda. El branch se resuelve en la etapa EX:

```
Loop:   LW R10,0(R1)
        ADD R4,R10,R2
        SW 0(R1),R4
        SUBI R1,R1,#4
        BNEZ R1,Loop
```

Dado que no dan información acerca de la toma de decisiones respecto del salto (ni siquiera que el salto se predice como no tomado), asumo que se hace flush del pipeline.



14) Explique porqué un Register File no introduce un riesgo estructural en un procesador MIPS con pipeline de 5 etapas.

Esto no sucede pues se implementa al Reg. File de tal manera que en el primer semiciclo se permita escribir (proveniente la escritura desde el WB) y en el segundo, leer (proveniente desde el ID). Es importante que sea en ese orden, para evitar riesgos RAW. Dado que sólo se usa el Reg. File para estas dos etapas, y solucionado este problema, no presenta un riesgo estructural para el pipeline.

15- Suponer las siguientes frecuencias de saltos (como porcentaje del total de instrucciones):
Saltos condicionales 15 % (60 % son tomados)
Saltos incondicionales y llamadas 1 %
Estamos examinando un pipeline MIPS de 4 etapas (IF-ID-EX/MEM-WB) donde los saltos incondicionales son resueltos al final del segundo ciclo, y los condicionales al final del tercer ciclo. Asumiendo que la primera etapa puede hacerse independientemente del resultado del salto, e ignorando otras situaciones de detenimiento del pipeline, ¿cuánto más rápido sería la máquina sin riesgos de salto (branch hazards)?

Separamos los saltos condicionales entre tomados y no tomados:9 % del total son saltos condicionales tomados, y 6 % son no tomados.
El caso de los saltos condicionales no tomados, no se pierde ningún ciclo. Los tomados nos

hacen perder dos ciclos. Los saltos incondicionales nos hacen perder 1 ciclo.
Si no tuvieramos los riesgos de control, 100 instrucciones se ejecutarían en 100 ciclos (quitando el tema de la primera instrucción, despreciable si en vez de 100 te pongo 100000000). Con estos riesgos, tardamos en promedio:

$$T_{100} = 100 + 2 \times 9 + 1 \times 1 = 119$$

Por lo tanto, sin los hazards, sería 19 % más rápida la máquina.

16- Calcular la penalidad de branch para un BTB (Branch Target Buffer) con estos parámetros: 60 % de branches son tomados, el BTB tiene una tasa de hit de 90 % y 95 % de precisión en la predicción.
BTB hit y correcta predicción tomado - penalidad de 2 ciclos
BTB hit y correcta predicción no tomado - penalidad de 0 ciclos
BTB hit e incorrecta predicción tomado - penalidad de 8 ciclos
BTB hit e incorrecta predicción no tomado - penalidad de 8 ciclos
BTB miss y tomado - penalidad de 8 ciclos
BTB miss y no tomado - penalidad de 0 ciclos

Tenemos los datos respecto del 100 % de las instrucciones. Tendremos que poner en práctica un par de contenidos de Probabilidad (fórmula de Probabilidad Total).

$$CiclosdePenalidad = P(Tomado)Penalidad_{Tomado} + P(NoTomado)Penalidad_{NoTomado}$$

$$P(Tomado) = 0{,}6$$
$$P(NoTomado) = 0{,}4$$

$$Penalidad_{Tomado} = P(Hit)Penalidad_{Tomado-Hit} + P(Miss)Penalidad_{Tomado-Miss}$$

$$P(Hit) = 0{,}9$$
$$P(Miss) = 0{,}1$$

$$Penalidad_{Tomado-Hit} = P(PrediccionCorrecta)Penalidad_{Tomado-PrediccionCorrecta}$$
$$+P(PrediccionIncorrecta)Penalidad_{Tomado-PrediccionIncorrecta}$$
$$P(PrediccionCorrecta) = 0{,}95$$
$$P(PrediccionIncorrecta) = 0{,}05$$

$$Penalidad_{Tomado-PrediccionCorrecta} = "Predecir - Correctamente - Tomado" = 2ciclos$$

$$Penalidad_{Tomado-PrediccionIncorrecta} = "Predecir - Incorrectamente - No - Tomado" = 8ciclos$$
$$Penalidad_{Tomado-Miss} = "Miss - y - Tomado" = 8ciclos$$

$$\Rightarrow Penalidad_{Tomado} = 0{,}9(0{,}95 * 2 + 0{,}05 * 8) + 0{,}1 * 8 = 3{,}1$$

$$Penalidad_{NoTomado} = P(Hit)Penalidad_{NoTomado-Hit} + P(Miss)Penalidad_{NoTomado-Miss}$$

$$P(Hit) = 0{,}9$$

$$P(Miss) = 0{,}1$$

$$Penalidad_{NoTomado-Hit} = P(PrediccionCorrecta)Penalidad_{NoTomado-PrediccionCorrecta}$$

$$+P(PrediccionIncorrecta)Penalidad_{NoTomado-PrediccionIncorrecta}$$

$$P(PrediccionCorrecta) = 0{,}95$$

$$P(PrediccionIncorrecta) = 0{,}05$$

$$Penalidad_{NoTomado-PrediccionCorrecta} = "Predecir - Correctamente - No - Tomado" = 0ciclos$$

$$Penalidad_{NoTomado-PrediccionIncorrecta} = "Predecir - Incorrectamente - Tomado" = 8ciclos$$

$$Penalidad_{NoTomado-Miss} = "Miss - y - NoTomado" = 0ciclos$$

$$\Rightarrow Penalidad_{Tomado} = 0{,}9(0{,}95 * 0 + 0{,}05 * 8) + 0{,}1 * 0 = 0{,}36$$

$$\Rightarrow CiclosdePenalidad = 0{,}6 * 3{,}1 + 0{,}4 * 0{,}36 = 2{,}004$$

17) Dibujar el datapath completo para la arquitectura MIPS pipeline de 5 etapas. Incluir el forwarding ALU-ALU y Memoria-ALU.

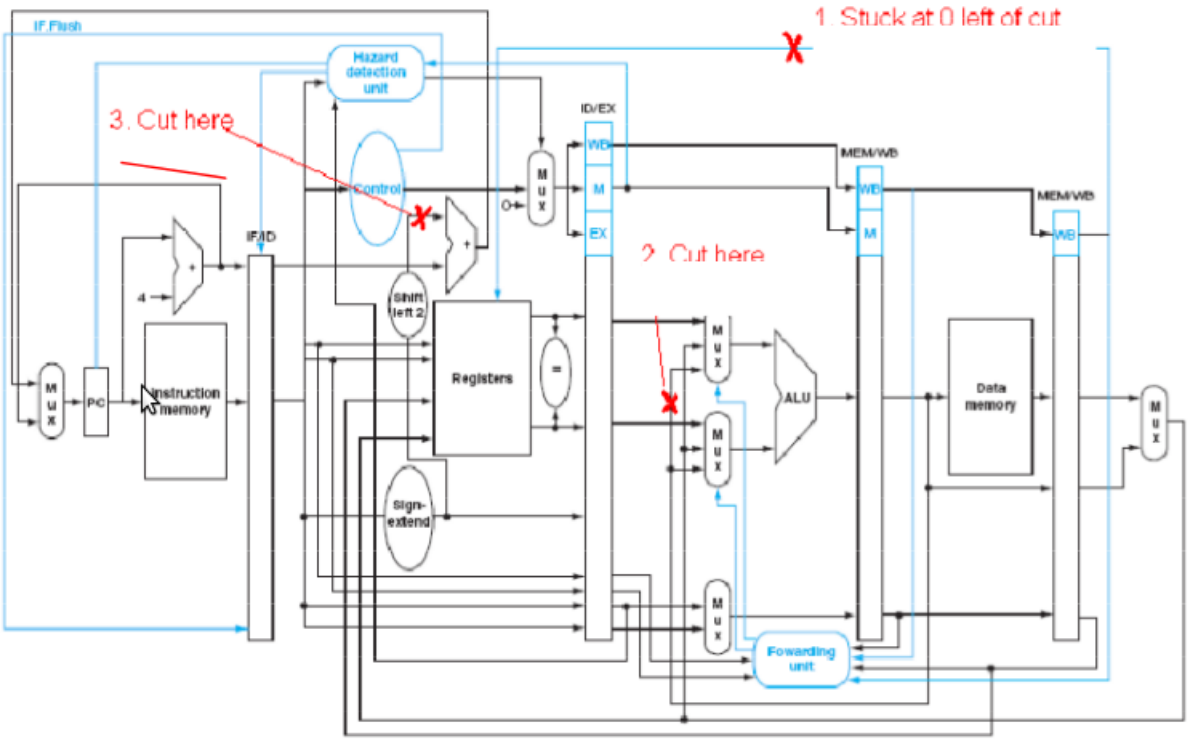
Bueno, otra vez pidiendo lo mismo, así que a estudiarlo porque hay chances de que lo tome!

18- Explique los dos significados del término SIMD, uno como el usado en la computadora Illiac IV (1964) y otro como el usado en las extensiones MMX/SSE/SSE-2 de Intel.

Para las extensiones MMX o SSE el SIMD es el tipo de taxonomía Flynn que permite trabajar con un gran flujo de datos en simultáneo para extender el set de instrucciones para realizar funciones específicas (como la extensión multimedia).
Lo otro, ni idea.

19) Para el datapath MIPS dado a continuación, varios links han sido numerados y marcados con X. Individualmente para cada uno los mismos:
. Describa con palabras las consecuencias negativas para el funcionamiento del procesador de cortarlo en el punto marcado.
. De una secuencia de instrucciones reales MIPS tal que asumiendo el camino cortado hará que la secuencia falle en su ejecución.
. De una secuencia de instrucciones reales MIPS tal que aún con el camino cortado

la secuencia funcionará correctamente.



1. Se corta el cable que permite indicarle al Register file que se quiere escribir un registro. Por lo tanto, no podemos darle nuevos valores a ningún registro ¿Qué más negativo que eso? Queremos una secuencia que falle al hacer esto, es muy simple:

```
addu r1, r2, r3
```

Queremos una que no haga WB, básicamente, para que siga funcionando:

```
sw  0(r2), r3
```

2. Cortamos el cable que va al multiplexor del primer operando de la ALU, que permite hacer Fowarding ALU-ALU. De esta manera, no podemos manejar los riesgos RAW de operaciones que necesiten que la ALU use consecutivamente un mismo operando que haya sido modificado en la instrucción anterior. Ejemplo que permite mostrar que esto no funciona correctamente:

```
addu r3, r2, r1
addu r5, r3, r4
```

Ejemplo que permite ver que esto siga funcionando correctamente:

```
addu r3, r2, r1
addu r4, r2, r5
```

3. Al cortar ese cable, no podemos realizar ningún Branch, puesto que no va a andar correctamente el salto. Ejemplo de esto andando mal:

```
loop:   add  r1, r2, r3
        subi r2, 10
        bne  r1, loop
```

No sabemos como responderá tal programa, pero probablemente no haga el salto esperado. Un ejemplo en el que siga funcionando: cualquiera en el que no haya un branch.

20) Dibuje un diagrama en bloques detallado de un Register File para un procesador MIPS con pipeline de 5 etapas.

Mismo que el ejercicio 1, así que a tenerlo bien en cuenta porque parece que le gusta.

6. Predictores de Salto

1- Describir cómo funciona un predictor de saltos de dos niveles y por qué un esquema así puede ser efectivo.

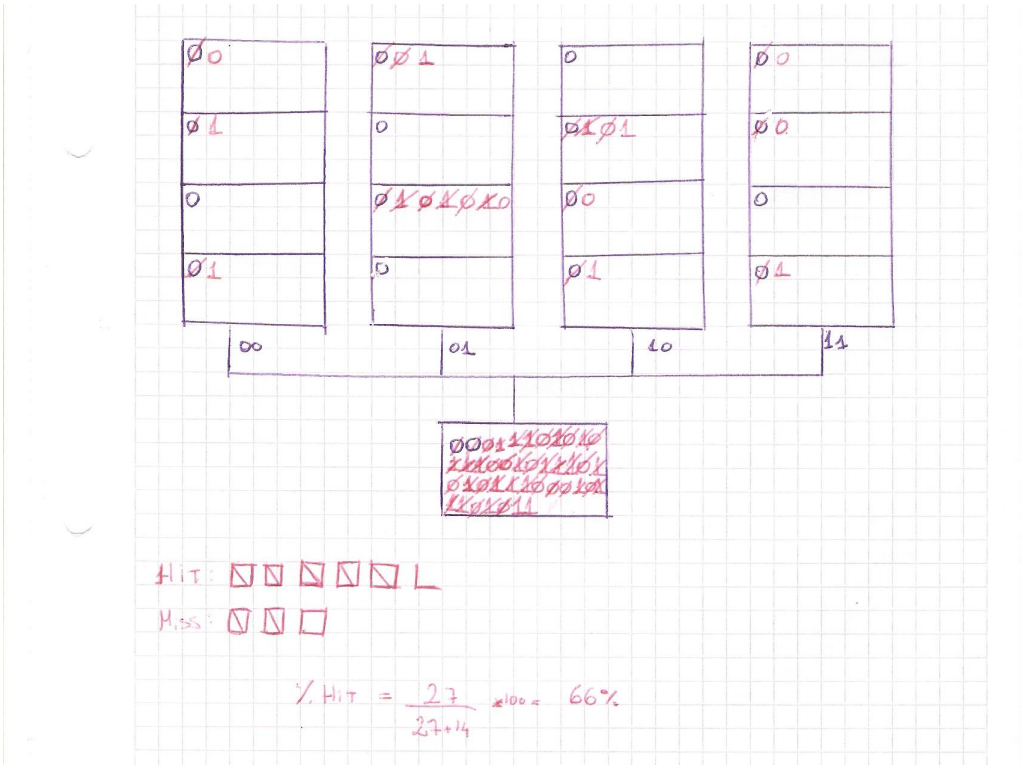
Un predictor de salto de dos niveles es uno que contiene una cantidad determinados de bits como historia global del salto (se va actualizando dependiendo del último salto, indpendientemente de qué tipo era), con el cual se elije una tabla de historia local, indexada por el PC del salto (se toman los n bits menos significativos) (se tienen $2^{cantbitsh.glob}$ tablas). Dependiendo del estado de la historia local (que puede ser un predictor local de 1 o dos bits, o más) se predecirá el salto como tomado o no, luego actualizando con el caso verdadero. Un esquema así puede funcionar correctamente, pues se basa en patrones (historia global) y al mismo tiempo en el caso particular de cómo funciona un determinado branch ante tales patrones, por lo que en caso de bucles puede llegar a generar una tasa de miss muy pequeña.

2- Dibuje el esquema de un predictor de saltos de 2 niveles y explique porque puede lograr altas tasas de aciertos en la prediccción.

La explicación ya fue hecha en 1), y el dibujo lo hago más adelante en otro ejercicio.

3) Realice un esquema de un predictor de saltos en dos niveles con 2 bits de historia global y 1 de historia local. Porque que logra un buen funcionamiento este tipo de predictores. Calcule la tasa de predicción del predictor pedido para el código dado (código del problema 11 de Arquitecturas). Suponer que la primera vez de cada branch la predicción es NT.

La explicación ya fue dada, ahora pongo el esquema con el recorrido hecho sobre las instrucciones (es un trabajo un tanto laborioso, y muy probable de generarse una pifia en un examen).



4) Considere agregar la predicción de branch al pipeline MIPS de 5 etapas. Asuma que el pipeline no utiliza el esquema de salto demorado. Asuma que la predicción de saltos dinámica se realiza en la etapa de ID y que los saltos se resuelven en la etapa de MEM del pipeline. Conteste las preguntas siguientes y proporcione las explicaciones para cada una. Proporcione cualquier diagrama que usted piense será provechoso para explicar su respuesta.
a) ¿Cuántos ciclos se pierden si un branch se predice correctamente no tomado?
b) ¿Cuántos ciclos se pierden si un branch se predice correctamente tomado?
c) ¿Cuántas ciclos se pierden si un branch se predice incorrectamente no tomado?
d) ¿Cuántas ciclos se pierden si un branch se predice incorrectamente tomado?

a) Si se predice correctamente como no tomado, dado que se siguió cargando la siguiente instrucción, mientras en ID se hacía la predicción, no se perdió ningún ciclo.

b) Si se predice correctamente como tomado, dado que se tuvo que sacar una instrucción siguiente que entró incorrectamente al pipeline, se perderá un ciclo.

c) Dado que se predice recién en la segunda etapa, y en la tercera conocemos el verdadero resultado, perdimos 2 ciclos.

d) Igual al anterior, pero se tuvo que primero vaciar el pipeline, por lo que la instrucción eliminada (nuevamente) recién estará en etapa de Fetch, mientras que en c) se encuentra en etapa de ID.

7. Tópicos especiales

1- Procesamiento multihilo (multithreading).
a) Políticas de procesamiento multihilo.
b) Desperdicios vertical y horizontal.
c) Para cada uno de los siguientes elementos decir si hay uno individual para cada hilo o hay uno solo compartido entre todos los hilos.
I) Memoria de código.
II) Memoria de datos.
III) Estructura de Stack.
IV) Registros Generales.
V) Contador de programa.
VI) Memoria cache.
Justificar.

a) Es un poco escueta la pregunta.. supongo que debe estar preguntando que se mencionen políticas de procesamiento multihilo. Para esto tenemos:

- CMP (Multiprocesador en el Chip): Se tienen varios procesadores, cada uno trabajando en un proceso distinto
- CMT (Multihilo de grano grueso): Un procesador va ejecutando de a un hilo por vez, y cambia de hilo en cuanto el hilo ejecutado llega a una penalidad (que puede ser un miss en caché, por ejemplo). Ejemplo: IBM Northstar.
- FMT (Multihilo de grano fino): Un procesador va cambiando de un hilo a otro cada cierta cantidad de instrucciones (por ejemplo, podría ser cada cierta cantidad de tiempo sino). Para este ejemplo, iría ejecutando una instrucción de cada thread por vez. Ejemplo: Denelcor HEP (falló, comercialmente hablando).
- SMT (Multihilo simultáneo): Se ejecutan instrucciones de distintos threads en un mismo ciclo de reloj, en paralelo, sin necesidad de duplicar las unidades funcionales. El procesador coordina las distintas instrucciones que se ejecutan en paralelo sobre una misma unidad funcional. Ejemplo: Pentium 4 o Alpha 21464.

b) Los desperdicios horizontales en un procesador superescalar se dan cuando no se puede utilizar todo el ancho del superescalar debido a la dependencia de datos entre instrucciones (hazzards de datos). Los desperdicios verticales se dan cuando toda una unidad del superescalar no está siendo utilizada, probablemente porque se haya bloqueado por alguna razón (ej: hubo un page fault, y bueno.. hay que esperar). ¿Cómo se lo puede mejorar? Si se permite utilizar procesamiento multihilo, aunque sea de grano fino o grueso, habrá una mejora, pues se pueden ir reutilizando los elementos del superescalar que no estén sieno utilizados. Si se utiliza un multihilo simultáneo, la cosa mejora aún más.
(Leer en el resumen de Nico)

c) I) La memoria de instrucciones es una sola.
II) La memoria de datos es una sola (y más le vale, si justamente la idea es que se puedan comunicar los threads)

III) No se si se refiere a lo que están en memoria de stack, si se subdivide por proceso. Si es así, no, es todo lo mismo. Si se refiere a que cada función de cada proceso tenga su propio stack, entonces si, cada uno tiene el suyo, pero es sobre la misma porción de memoria.
IV) Los registros siguen siendo los mismos (El Reg file sigue siendo el mismo).
V) El PC se duplica, y se tiene uno por cada thread (Y.. sino se complica saber por dónde ir ejecutando cada uno).
VI) La caché sigue siendo la misma.. a lo sumo se le aumentará el tamaño para que funcione mejor.

2- Explique la principal diferencia entre una arquitectura superescalar y una VLIW.

Que ganas de romper la pelotas que tiene este tipo...
VLIW: Very Long Instruction Word. En esta arquitectura, el compilador genera una palabra que contiene varias instrucciones que pueden ejecutarse en paralelo, rellenando espacios con nops en caso de no encontrar algo que permita cerrar la cosa. En una arquitectura superescalar, quien verifica si las cosas se puedan hacer o no en paralelo es el hardware, mandando a cada vía dependiendo del caso. Entonces, la principal diferencia radica en que en la VLIW quien analiza el paralelismo es el compilador (software), mientras que en una superescalar lo hace el hardware.

3- Explique el funcionamiento básico de un procesador superescalar que utiliza:
a- Scoreboarding (tablero de resultados)
b- Algoritmo de Tomasulo

a) No encuentro en ningún resumen, ni en las diapositivas, qué es esto.

b) El algoritmo de Tomasulo implementa una organización particular para el data path de un superescalar. Se tienen Buffers y control distribuido con unidades funcionales que accedan a memoria (Estos buffers permiten poder ir accediendo a memoria de manera ordenada, pues la memoria física no se duplica, por lo que hay que tener cuidado al usar estas cosas). De esta manera, se logra escribir en orden preciso, y leer el resultado correcto (cada buffer debe darse cuenta de cuál es el resultado que espera). A estos buffers se los llaman estaciones de reserva. El acceso a registros es reemplazado por valores o punteros a estaciones de reserva (esto permite elimiar riesgos RAW). Esto es muy imporante, porque en la etapa de ID se obtienen los operandos, pero en el pipeline tradicional, si estos no estaban listos, se usaba un ciclo de stall para evitar problemas con la etapa de EX. Entonces, en la estación de reserva correspondiente se tiene guardado un puntero de quién (otra RS) es el responsable de generar el dato faltante. Para que esto funcione, las estaciones de reserva se comunican por un bus único, por lo que con hacer un simple broadcast, el dato de una RS le llega a todas las demás. Para resolver que instrucción va a que RS, se utiliza una cola de instrucciones, que luego deriva a cada instrucción. Aparte: los Load y Store son tratados como unidades funcionales.

8. Verdadero o Falso

1) Verdadero o falso, justificar.
a) El CPI promedio de un procesador MIPS con pipeline es 1 porque se ejecuta una instrucción por ciclo.
b) La memoria virtual es el área de swap del disco rígido.
c) Las políticas de reemplazo en memoria cache se implementan por el sistema operativo.
d) El registro de instrucciones de una CPU es invisible a la arquitectura del conjunto de instrucciones.

a) Falso. Aunque efectivamente el CPI es 1, no es porque la instrucción se ejecute en 1 ciclo. La instrucción se ejecuta en varios ciclos, pero al estar ejecutando varias instrucciones a la vez (en distintas etapas), se logra que el CPI tienda a 1 (quitando el tema de accesos a memoria, que si hay miss en una cache puede tardar más, etc...).

b) Falso. La memoria virtual es toda la memoria con la que cuenta el programador a usar cuando programa. Este depende del set de instrucciones. Si una dirección de memoria está activa (en memoria física), o es necesario ir al área de Swap a buscarlo, es transparente para el programa. La idea de memoria virtual nos permite abstraernos de la máquina particular en la que nos encontramos.

c) Falso. Las políticas de reemplazo son propias de la caché. Por ejemplo, la LRU implementa una pila para saber cuál fue es el valor que no se ha utilizado últimamente.

d) Verdadero. Este es un registro propio de la implementación del set de instrucciones, pero no es utilizable por este.

2) Verdadero o falso, justificar.
a) La memoria virtual paginada es transparente a la arquitectura de programación.
b) En una cache direccionada por direcciones físicas la TLB es accedida antes que la cache.
c) Un procesador superescalar pertenece a la categoría SISD de la taxonomía de Flynn.
d) El stack pointer del procesador MIPS es uno de los registros del regiter file.

a) Verdadero. Al estar la memoria virtual separada por un rango específico, el programador no tiene que estar trabajando con segment+offset. Simplemente accede a una dirección de memoria y listo. Si se trata de memoria física real, de swap, o no está siquiera implementada la memoria virtual (y si hay o no más de un proceso ejecutándose) al programador le resbala.

b) Verdadero. Es necesario antes de ir a la Caché ir a visitar a la amiga TLB para que nos de una dirección física.. sino no sirve de mucho la caché. Esto se puede apreciar viendo el diagrama de bloques con la Caché y la TLB (clase 4 de las diapositivas).

c) Esta pregunta es medio tramposa. Si el superescalar es para un único hilo, la respuesta

es: verdadero, porque hay un único flujo de programa (único hilo ejecutándose a la vez), y un único PC.

d) Em.. técnicamente es falso. No hay un registro (como sí pasaba con el r14 en ARC) que guarde automáticamente el stack pointer. Nosotros tomamos la responsabilidad de ir guardando el global pointer y el frame pointer.. así que no sabría decir exactamente, porque siempre habrá un registro que lo tenga guardado.

3) Verdadero o falso, justificar.
a) Las extensiones multimedia al conjunto de instrucciones tipo SSE pertenecen a la categoría MIMD de la taxonomía de Flynn.
b) La TLB es una tabla implementada en la memoria principal .
c) Un procesador SMT tiene espacios de direcciones de memoria individuales para cada hilo.
d) El contador de programa (PC) del procesador MIPS puede ser usado en instrucciones aritméticas del conjunto de instrucciones.
e) El modelo de las 3C clasifica a los misses de bloques de memoria que mapean en el mismo conjunto como "misses de conflicto".
f) El forwarding elimina la necesidad de stall en la siguiente secuencia:

```
lw r2,0(r4)
addi r8,r2,#2
```

a) Falso, pertenecen a la clasificación SIMD, pues es un único flujo de instrucciones el que trabaja sobre un mayor flujo de datos.

b) Falso, es una memoria Cache, así que está aparte de la memoria principal (y necesariamente antes). La pregunta sería 'Verdadera' si hablara sobre la tabla de traducción de página.

c) Emmm esto depende. Si está implementada la memoria virtual, de tal manera que se pueda manejar este tipo de cosas, entonces si (Verdadero). Y espero que justamente lo esté.

d) Falso. El PC, así como el IR, son registros de implementación. Eso quiere decir, que no tienen que ver directamente con el set de instrucciones. Podría estar implementado de tal manera que no se necesite el PC. Por esto, es invisible al programador, y por ende, no se lo puede usar en una operación, o cualquier otra cosa.

e) Falso. No necesariamente. Puede ser que si, pero si un cierto bloque nunca fue utilizado y se lo llama por primera vez, será un miss compulsivo. Incluso, podría también ser de capacidad (si el miss se produciera también en una FA de mismo tamaño a la de estudio).

f) Falso. Elimina uno de los stalls necesarios (el de la etapa de WB), pero dado que tenemos que esperar hasta la etapa de MEM, igualmente habrá aunque sea un ciclo de stall inevitable con un pipeline de 5 etapas.

4) Conteste verdadero (V) o falso (F)
__ **Una cache LRU de 2 vias siempre produce menos misses que una cache de con**

los mismos parámetros pero de correspondencia directa.
__Una cache LRU de una dada organización siempre produce menos misses que una exactamente igual pero mas pequeña.
__Una cache FIFO de una dada organización siempre produce menos misses que una exactamente igual pero mas pequeña.
__ El pipeline disminuye la latencia para la ejecución de las instrucciones.
__ En un pipeline de 5 etapas donde cada etapa requiere un ciclo de reloj el CPI ideal de las instrucciones es 5.
__ En una cache direccionada por direcciones físicas antes de acceder a la cache de debe acceder a la TLB.
__ En un sistema de memoria virtual la memoria física es la RAM y la virtual es el área de swap en el disco.
__ La memoria virtual paginada es invisible a la arquitectura del conjunto de instrucciones.

Como no dice que justifique, y deja el espacio para poner una simple letra, respondo así de una:

a) Falso (ej: hilo que accede a 1.5 tamaño de la cache, en la 2WFA da 100 % de miss, y en la otra no).

b) Verdadero (y.. si tengo más lugar, seguramente entran más cosas)

c) Falso (creo que depende de los accesos que se vayan a hacer) -¿Revisar

d) Verdadero (porque entra una mientras se ejecuta se empieza a ejecutar la siguiente, no se espera a que una termine).

e) Falso (es 1)

f) Verdadero (y sino como queres direccionar?)

g) Eeeeeeeeehhhhh (sale volando). Falso, toda la memoria es virtual.

h) Verdadero (ya se explicó 10 millones de veces).

5) Verdadero o Falso (justificar)
a- Una cache DM nunca puede tener una mejor tasa miss que una LRU FA del mismo tamaño y para el mismo patrón de referencias.
b- Aun si hubiera más memoria física que la virtual utilizada por todas las aplicaciones simultáneamente, el esquema de memoria virtual seguiría siendo muy útil.
c- Los registros pipeline MIPS son parte de la arquitectura de programación.
d- El concepto fundamental de un procesador RISC es su arquitectura load/store.
e- El modelo de las 3C clasifica los misses individualmente.
f- Un procesador superescalar es del tipo SISD de la taxonomía de Flynn.

a) Super falso. Ej: se ejecuta un hilo de modelo de lazo simple, de una posición más de memoria que la soportada por las Cache. La cache FA dara 100 % de miss rate, mientras que la otra dara mucho menos, pues solo se va poniendo y sacando dos elementos (el primero y el ultimo).

b) Verdadero. El esquema de memoria virtual permite simular la memoria que se tiene, que viene a ser la máxima soportable por el set de instrucciones. Tener más no te va a permitir acceder a más memoria, porque el set no te lo va a permitir, pero puede servir para el manejo de procesos separados (multithreading) para ir asignándole distintas porciones de memoria a dos procesos que piden por la misma dirección virtual.

c) Falso. Son registros de implementación. Si no se implementara el pipeline, no sería cosa de cambiar el set. Como ya fue explicado, son registros internos de la implementación y tienen un uso específico, así que está bien que no estén disponibles.

d) En gran parte Verdadero. La idea es limitar los accesos a memoria solo para lecturas y escrituras, y de esta manera, se hacen más simples las instrucciones (aunque tenga como contra que se necesiten más instrucciones para hacer lo mismo, y un set de instrucciones más grande), lo cual también es importante para una arquitectura RISC.

e) Falso. Se tiene en cuenta el contexto en el que se dieron, pues se toma en cuenta si un miss se produce por ser un bloque nuevo llamado, si se dio porque simplemente no había más lugar, o si la secuencia de llamadas produjo un miss dada la organización de la cache.

f) Como ya fue mencionado, si se mantiene un único hilo de ejecución, es verdadero.

6) Verdadero o falso, justificar.
a) La arquitectura intel X86 actual al ser compatible hacia atrás y anterior a las arquitecturas load/store no se beneficia de la ventajas de estas últimas.
b) Un procesador superescalar de 4 vías tiene 4 stacks.
c) El contador de programa (PC) del procesador MIPS puede ser incrementado mediante una instrucción aritmética del conjunto de instrucciones.
d) Los MIPS (millones de instrucciones por segundo) no indican cuan rápida es una computadora y no se relacionan con el tiempo de ejecución de las aplicaciones.
e) Un procesador de múltiples núcleos con cache compartida el del tipo MIMD de la taxonomía de Flynn.

a) No tengo ni la más puta idea.

b) Falso, porque el stack están en memoria, y esta no se duplica. Como mucho podrían tenerse distintos punteros a ella.

c) Falso. Ya fue explicado.

d) Verdadero. Una cosa no tiene nada qe ver con la otra. Tiene una cierta relación porque al fin y al cabo trabaja con campos que indican el tiempo de CPU, pero no con la velocidad

(es un elemento de chamuyo cuando te venden una computadora).

e) Verdadero. No hay mucho mas para decir, que el hecho que la arquitectura mencionada justamente implica que se tiene más de un procesador, que cada uno corre un proceso (programa) distinto, y cada uno tiene un flujo de datos por separado.

7) Verdadero o falso, justificar.
a) El tamaño del código de una aplicación es una medida estática.
b) Un procesador SMT de 4 vías tiene 4 stacks.
c) El contador de programa (PC) del procesador MIPS puede ser incrementado mediante una instrucción aritmética del conjunto de instrucciones.
d) Los MIPS (millones de instrucciones por segundo) son útiles para medir desempeño.
e) Un procesador superescalar el del tipo SISD de la taxonomía de Flynn.

a) Falso. Esto depende de la arquitectura del set de instrucciones. Si en particular es del tipo Reg-Reg, la respuesta sería verdadero, dependiendo directamente de la cantidad de instrucciones que haya en el código.

b) Falso, ya fue explicado más arriba.

c) Otra vez? Falso

d) Falso, explicado arriba. Es puro chamuyo esta cifra.

e) Otra vez... mejor acordarse de esta.

8) Verdadero o Falso (justificar)
a- Un procesador con coprocesador de punto flotante ejecuta instrucciones a una tasa más lenta que sin coprocesador de punto flotante.
b- Un procesador superescalar de 2 vías puede ejecutar dos flujos de instrucciones en paralelo.
c- Las supercomputadoras actuales basan su desempeño de la aplicación de pipelines muy profundos.
d- La memoria virtual es el área de swap del disco rígido.
e- Las extensiones multimedia al conjunto de instrucciones tipo SSE pertenecen a la categoría MIMD de la taxonomía de Flynn.
f-El modelo de las 3C clasifica a los misses de bloques de memoria que mapean en el mismo conjunto como "misses de conflicto".
g- La arquitectura MIPS 32 posee instrucciones de ejecución condicional.

a) Que se yo... si el coprocesador es lento supongo que si (?) Revisar esta

b) Verdadero. Esto incluso mejora los problemas de desperdicios verticales y horizontales.

c) Verdadero, aunque definamos 'profundo'. Supongo que es que se subdividen en muchas

etapas. Si es así, hoy en día el pipeline debe ser utilizado en base a más de 20 etapas.

d) Falso. Explicado ya.

e) Ya está respondido.

f) Falso, otra vez.

g) Falso. Se refiere a operaciones ALU que permiten evaluar condiciones y operar a partir de éstas. Es un método por Harware para reducción de saltos.

9) Verdadero o Falso (justificar)
a- La política de reemplazo de la cache de nivel 1 es manejado por el SO.
b- El TARGET de un jump en MIPS es independiente de la dirección del jump (Nota del que responde: debe referirse a la dirección de la instrucción de jump).
c- Los registros SP y FP en MIPS son registros dedicados de la arquitectura del conjunto de instrucciones.
d- Una arquitectura memoria-memoria produce ejecutables con menos instrucciones que una arquitectura Load/Store.
e- Un procesador con coprocesador de punto flotante ejecuta instrucciones a una tasa más lenta que sin coprocesador de punto flotante.
f- Un procesador superescalar de 2 vías puede ejecutar dos flujos de instrucciones.
g- Las supercomputadoras actuales basan su desempeño de la aplicación de pipelines muy profundos.

a) Falso, re falso (ya explicado en otra pregunta).

b) Falso, a la dirección Target se le suma el valor del PC.

c) En teoría verdadero. Como poder se pueden usar para otra cosa, pero rompés todo, porque hay una cuestión de convenciones en el medio.

d) Verdadero, porque se necesitan menos instrucciones para hacer lo mismo (ej, un simple add, no es necesario traer los operandos desde memoria, luego hacer la operacion y al final guardar el resultado, sino simplemente hacer la operacion y listo).

e) ni idea, hay que averiguarlo

f) Verdadero, justamente es una utilidad muy grande del superescalar (se aprovecha mucho más su potencial).

g) Ya fue respondido esto.

10) Verdadero o Falso (justificar)
a- Una cache FA FIFO nunca puede tener una mejor tasa miss que otra FA FIFO de mayor tamaño y para el mismo patrón de referencias.

b- Una cache de 2 vías LRU nunca puede tener una mejor tasa miss que otra de 2 vías LRU de mayor tamaño y para el mismo patrón de referencias.
c- Una cache FA Random nunca puede tener una mejor tasa miss que otra FA Random de mayor tamaño y para el mismo patrón de referencias.
d- Aun si hubiera más memoria física que la virtual utilizada por todas las aplicaciones simultáneamente, el esquema de memoria virtual seguiría siendo muy útil.
e- Los registros IR y PC en MIPS son parte de la arquitectura de programación.
f- El concepto fundamental de un procesador RISC es que tiene pocas instrucciones.
g- El modelo de las 3C para misses en memoria es una taxonomía.
h- Un procesador superescalar es del tipo MIMD de la taxonomía de Flynn.

a) Verdadero, no se me ocurre ningun contraejemplo.

b) Verdadero, tampoco se me ocurre. Pensar si hay alguno

c) Falso, porque justamente es random, puede sacarte la que estas por leer en una, y en la otra no.

d) Verdadero, como ya fue explicado.

e) Falso, como ya fue explicado.

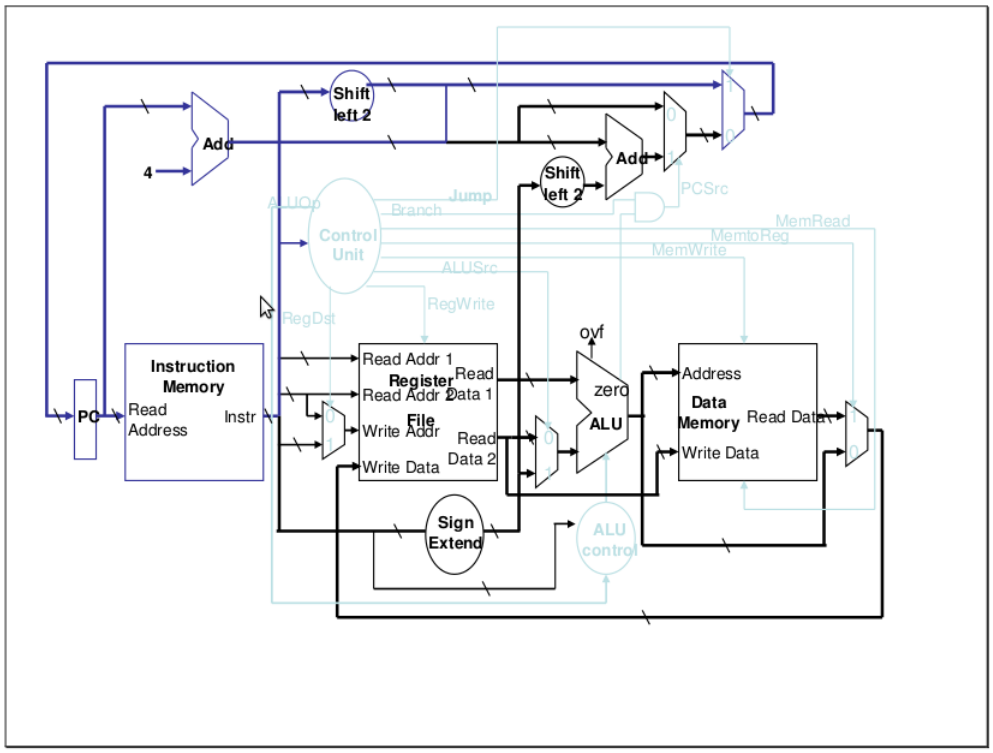
f) SUPER falso, como ya fue explicado en otra pregunta.

g) Verdadero, creo. Es una forma de clasificarlos para saber como conviene tratarlos, en caso de necesitarlo.

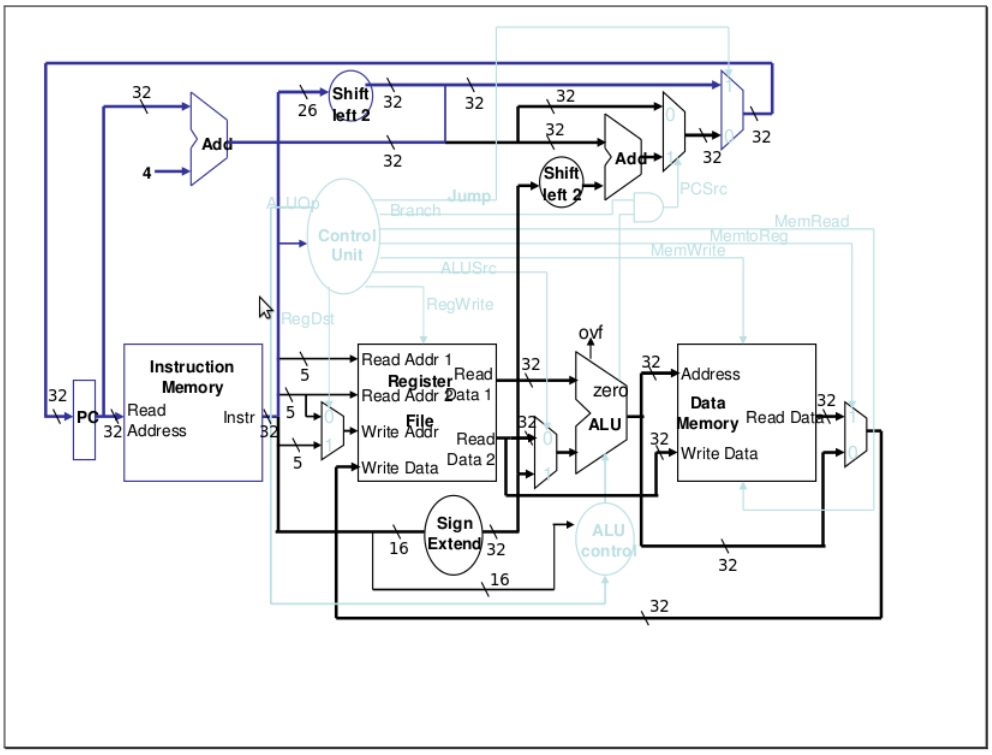
h) Falso. Algun multiprocesador que implemente un superescalar puede llegar a serlo, pero así de una te digo que no.

9. Camino de Datos

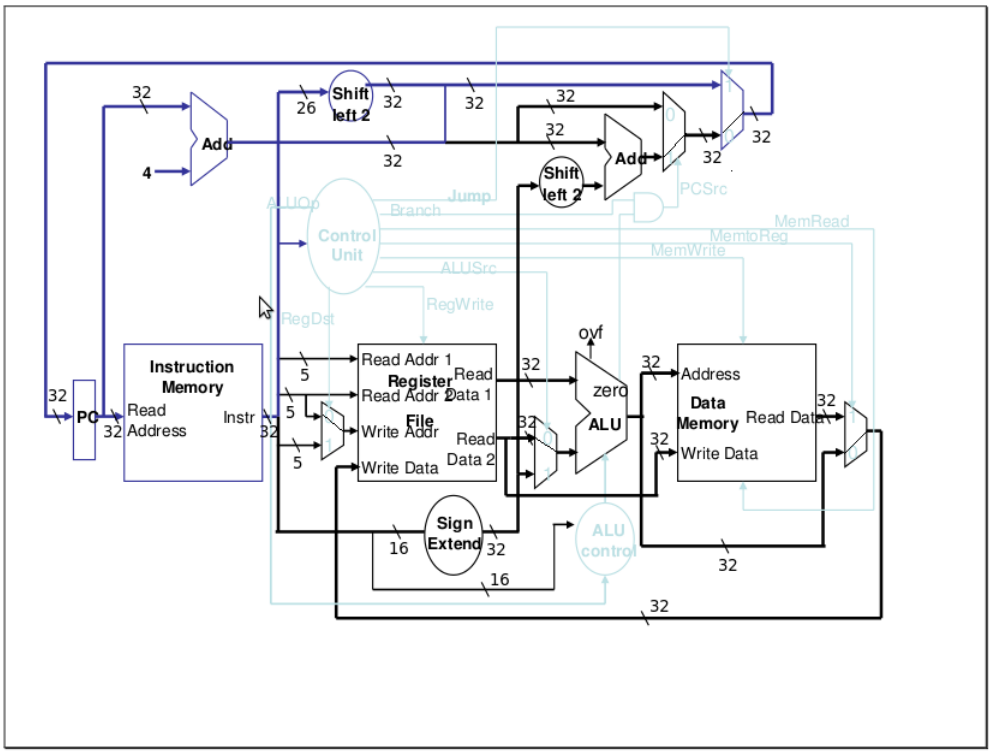
1) El siguiente dibujo corresponde a una implementación monociclo para un procesador MIPS. a) Indique sobre el dibujo junto a cada marca que cruza una conexión, la cantidad de líneas correspondiente. b) Dibuje dentro cada uno de los multiplexores mediante una línea, cual de las entradas se conecta con la salida, si se está ejecutando una instrucción "beq rs, rt, label".



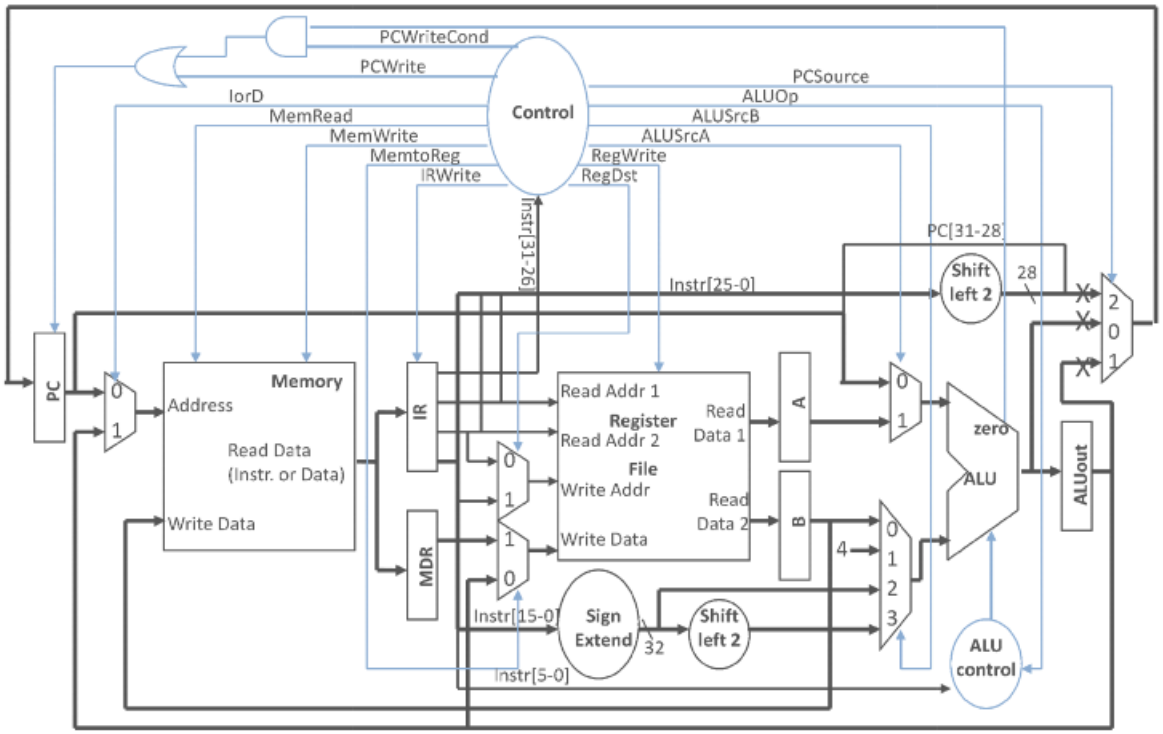
a)



b) Se hace lo siguiente asumiendo que el salto se toma:



2) La figura siguiente:



corresponde al datapathh multiciclo de un procesador MIPS. En la misma cada multiplexor ha sido numerado (1 al 6). Indicar en la siguiente tabla el número de entrada que debe conectarse con la salida de cada multiplexor para cada ciclo de ejecución de las siguientes instrucciones. Si en un determinado ciclo para un determinado multiplexor no interesa el valor de la salida, indicarlo con una X en el lugar correspondiente de la tabla:

	Ciclo				
MUX	IF	ID	EX	MEM	WB
MUX1					
MUX2					
MUX3					
MUX4					
MUX5					
MUX6					

Llenarla para las instrucciones add, load y beq.

No nos muestran cuál es cada multiplexor, pero asumo que el 1 es el que está cerca del PC, el 2 el que elige write Adr, el 3 el que elije wrt. data, el 4 que elije el srcA el 5 que elije el srcB y el 6 el que elije el dato a guardar. Así, lleno con la siguiente información:

a) Add (asumo que no vengo de ningún Branch):

	Ciclo				
MUX	IF	ID	EX	MEM	WB
MUX1	0	0	0	X	X
MUX2	X	X	X	X	1
MUX3	X	X	X	X	0
MUX4	X	X	1	X	0
MUX5	X	X	0	X	1
MUX6	X	X	X	X	2

El 0 y 1 en Mux 4 y Mux 5 en etapa WB se debe a que se está sumando el PC + 4 en ALU, y el MUX 6 selecciona el resultado de esa operación para ser el nuevo PC.

b)Load (asumo que no vengo de ningún Branch):

	Ciclo				
MUX	IF	ID	EX	MEM	WB
MUX1	0	0	0	X	X
MUX2	X	X	X	X	1
MUX3	X	X	X	X	0
MUX4	X	X	1	X	0
MUX5	X	X	2	X	1
MUX6	X	X	X	X	2

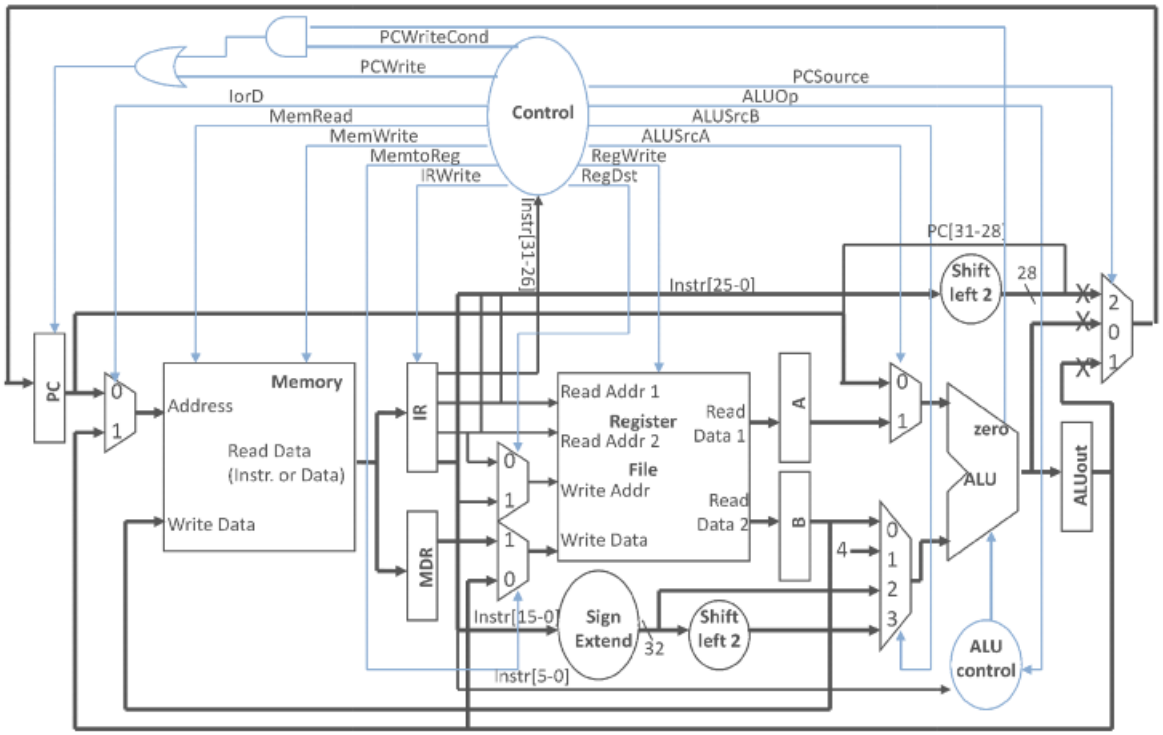
a) Branch condicional, beq (asumo que no vengo de ningún Branch):

	Ciclo				
MUX	IF	ID	EX	MEM	WB
MUX1	0	0	0	X	X
MUX2	X	X	X	X	X
MUX3	X	X	X	X	X
MUX4	X	X	1	X	X0
MUX5	X	X	0	X	X3
MUX6	X	X	X	X	1

Explico etapa WB: no hay registro a escribir, pero se está calculando la dirección e salto y si hay que saltar o no. Para esto el MUX4 selecciona el PC, y el MUX5 selecciona a la constante, multiplicada por 4 (shift 2), que indica el offset del salto. El resultado será retenido en el ALUOT, mientras se realiza la comparación, para saber si el branch se toma o no. Por lo tanto, MUX6 toma el valor desde el ALUout.

3- El siguiente dibujo corresponde a una implementación multiciclo para un procesador MIPS. Tres links (0, 1 y 2) de entrada a un multiplexor han sido marcados con X. Individualmente para cada uno los mismos: a. Describa con palabras las consecuencias negativas para el funcionamiento del procesador de cortar el link en el punto marcado. b. De una secuencia de instrucciones reales MIPS tal que asumiendo el camino cortado hará que la secuencia falle en su ejecución. c. De una

secuencia de instrucciones reales MIPS tal que aún con el camino cortado la secuencia funcionará correctamente. (Si la secuencia no existe decirlo explícitamente)



Cortar el 0: a. da el resultado de la ALU, para asignarlo al PC. Si se lo corta, la ALU no podrá hacer los cálculos de PC de la próxima instrucción (excepto los que transporta el cable 1, branchs conicionales; y el cable 2, saltos/jumps). Por lo tanto, habrá fallos con cualquier branch no condicional, y con el PC+4, el valor del PC queda corrupto.

b. add R1, R2, R3 add R4, R5, R6

c. loop: be loop

Cortar el 1: a. toma el valor del ALUout. Ésta guarda el resultado de ALU mientras se computa si el salto debe ser o no tomado (saltos condicionales). Fallará ante un salto condicional tomado, el valor del PC queda corrupto.

b. loop: be R1, R2, loop

c. add R1, R2, R3 add R4, R5, R6

Cortar el 2: a. toma el valor de la constante de 26 bits de las instrucciones tipo jump. Entonces, si hay algún jump, el valor del PC queda corrupto.

b. j SubRutina

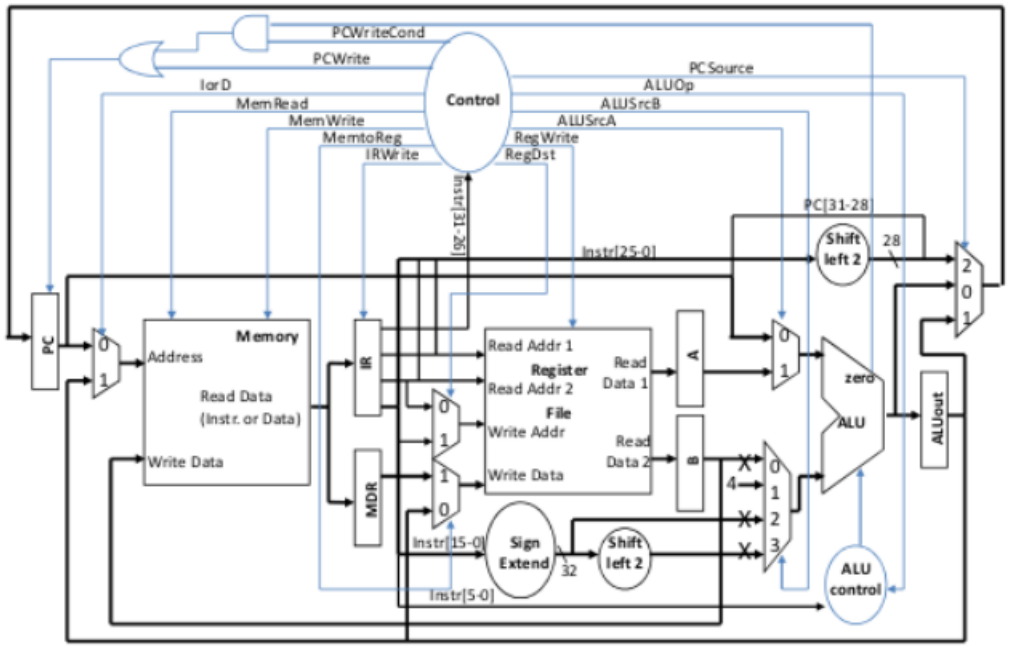
c. add R1, R2, R3 add R4, R5, R6

4- El siguiente dibujo corresponde a una implementación multiciclo para un procesador MIPS. Tres links (0, 2 y 3) de entrada a un multiplexor han sido marcados con X.
Individualmente para cada uno los mismos:

a. Describa con palabras las consecuencias negativas para el funcionamiento del procesador de cortar el link en el punto marcado.

b. De una secuencia de instrucciones reales MIPS tal que asumiendo el camino cortado hará que la secuencia falle en su ejecución.

c. De una secuencia de instrucciones reales MIPS tal que aún con el camino cortado la secuencia funcionará correctamente. (Si la secuencia no existe decirlo explícitamente).



El Multiplexor controla cuál es el operando B de alu. Posibles: 0: opera con registro fuente 2. 2: opera con constante. 3: opera con constante multiplicada por 4 (usado para cálculo de saltos).

Cortar el 0: a. las operaciones con segundo operando registro, darán resultados erróneos.

b. add R1, R2, R3

c. add R1, R2, 1

Cortar el 2: a. las operaciones con segundo operando constante, las operaciones inmediatas, darán resultados erróneos.

b. add R1, R2, 1

c. add R1, R2, R3

Cortar el 3: a. las instrcucciones que requieran un cálculo de Target no darán el resultado correcto. Estas son las intrucciones con Branch.

b. b loop

c. add R1, R2, R3 add R4, R5, R6

10. Finales

10.1. Final del 04/02/2013

1) Hacer el diagrama de un Datapath de un MIPS con Pipeline de 4 etapas.

Hacer un Diagrama del datapath pero con las cosas de ALU y MEM juntas, aludiendo que las instrucciones de ALU solo tardaran 4 ciclos (no pasaran por MEM) y que las de memoria (ld/st) tardaran un ciclo más, probablemente poniendo un ciclo de stall. Para esto se pueden tener dos unidades ALU, una mas potente y otra un simple sumador para indicar la posicion a leer/escribir, ademas de una señal que indique si se lee, si se escribe, o si se hace operacion de ALU, y un multiplexor que elija a partir de una or (entre la señal de lectura y la de escritura, si alguna vale 1, se toma la señal proveniente de memoria, sino, se toma la de la ALU), para mandar al WB. Esto fue sacado a partir de explicaciones en el Murdocca (aunque lo tiene hecho de otra manera, porque las etapas son algo distintas).

2) Extensión Multimedia:
a) ¿Qué es?
b) ¿Cómo se implementa?
c) ¿A qué taxonomía de Flynn pertenece?
d) Dar un ejemplo para MIPS.

a) La extensión Multimedia es una extensión al set de instrucciones, con instrucciones especializadas para poder manejar bloques de datos (o flujos enteros de datos). Con esto, puede (por ejemplo) manejarse sonido y video. Ejemplos: Extensión MMX de Intel a Pentium, y extensión AltiVec de Motorola a PowerPC.

b) Creo que la idea es que se tienen registros especiales (tipo 'vectores', que vienen a poder guardar mayor cantidad de datos), para asi poder manejar mayor información en simultáneo.

c) Pertenecen a la SIMD (Single instruction stream / multiple data stream) (porque se necesita tratar con mucha cantidad de datos de manera simultánea).

d) Después de buscar en Internet, encontré que hay uno que se llama MDMX (MIPS Digital Media eXtension).

3) Dar ventajas y limitaciones de Cache con index virtual y tag físico.

Ya se respondió esta pregunta (Sección de Caché).

10.2. Final del 18/02/2013

1) calcular qué tanto mejor es un procesador mips con pipeline de cuatro etapas frente a uno que tiene hazards de control, sabiendo que el 25 % de las instrucciones son de salto condicional (el 60 % se toman) y que el 3 % son saltos incondicionales. El pipeline resuelve los saltos condicionales en el tercer ciclo y los incondicionales

en el segundo y además sólo se aprovecha el fetch de la instrucción siguiente. (En voz alta aclaró que el pipeline funciona parecido a la estrategia de siempre elegir NT con la excepción de que sólo carga hasta IF).

La idea sería idéntica al ejercicio ya hecho en la sección de Pipeline, pero esta vez en los saltos condicionales no tomados se pierde un ciclo necesariamente (pues solo se aprovecha el ciclo de IF). Entonces, para un procesador sin hazards, 100 instrucciones se hacen en 100 ciclos (no tomando en cuenta el resto de los riesgos), mientras que para el que nos mencionan, 15 % de las instrucciones son saltos condicionales tomados (2 ciclos perdidos), 10 % de las instrucciones son saltos condicionales no tomados (1 ciclo perdido) y 3 % son saltos incondicionales (1 ciclo perdido):

$$T(100inst) = 100ck + 1ck \times 3 + 1ck \times 10 + 2ck \times 15 = 143ck$$

Por lo tanto, un procesador sin hazzards será un 43 % más veloz.

2) dibujar un esquema de hardware del BTB.

Hay que hacer el esquema de una cache con tag = PC (bah, dependiendo de la cantidad de bits a tomar), con el dato el destino (y posiblemente se le puede añadir el predictor para el salto).

3) dar un ejemplo de procesadores modernos para cada una de las categorías de la taxonomía de flynn. justificar.

- SISD (Single instruction stream / single data stream): Un procesador Pipeline, o un superescalar (de un solo proceso) son ejemplos de este, pues tenemos un solo flujo de datos y un único flujo de instrucciones (es un único proceso el que se está ejecutando).
- SIMD (Single instruction stream / multiple data stream): procesadores de extensión multimedia (ej: MMX o SSE de Intel para Pentium), porque a un mayor flujo de datos (o bloque de datos) debe trabajarlo en simultáneo (ejemplo, sonido), siendo un procesador especializado.
- MISD (Multiple instruction stream / single data stream): No existe, pues en general el procesamiento de datos en paralelo suelen ser más apropiadas para MIMD y SIMD. (A veces se usa el sistema para máquinas tolerantes a fallos, para detectar estos mismos y poder corregirlos).
- MIMD (Multiple instruction stream / multiple data stream): Una computadora con múltiples núcleos, pues cada uno puede trabajar en un proceso totalmente separado, con datos separados. Luego estos pueden estar o no acoplados.

10.3. Final del 25/02/2013

1) Ejercicio muy largo de Caché, que se puso a inventar los datos en el momento porque le faltaban muchas cosas. Ya no recuerdo el enunciado, pero era uno de esos que incluían TODO (todas las configuraciones posibles de Cachés, distintos niveles, etc...).

2) Hacer el diagrama del algoritmo de Tomasulo.
NO quería que lo expliques, sino que le hagas copy paste de la diapositiva... La explicación del

algoritmo está entre las secciones anteriores.

3) Topologías de comunicación de procesadores.
Ese era el enunciado, y aunque le pedimos que explique qué era lo que quería sobre eso, que además había dicho en clase que NO lo iba a tomar (clásico... hasta tenemos una grabación para probarlo), dijo que eso *Lo debíamos poder deducir nosotros*. Yo tiré diagramas de distintas configuraciones que recordaba, y nunca me dejó ver mi final, así que no sé si eso era lo que quería.

Recomendación: Estudiar todos los temas y más, porque hay fechas en las que va tranquilo, y toma lo que da, y más importante: los temas de la materia; pero hay fechas en las que toma cosas que no hacen a la materia (y son un detalle), o cosas que no explica directamente (o dice en clase que no lo va a tomar). Así que: estudiar mucho, y rezar.