

1. En un procesador **RISC-V con forwarding y predicción estática de NO TOMADO para los branches**, como el visto en clase, corre un programa para el que, con ciertas condiciones iniciales, se tiene la siguiente mezcla de instrucciones dinámicas (**IC**):
- **40%** Son **Operaciones aritméticas**, para las que
 - 1/8** parte tiene conflicto de datos para uno de sus operandos con un lw inmediato anterior
 - 1/4** parte tiene conflicto de datos para uno de sus operandos con otra aritmética tras-anterior
 - El resto no tiene conflictos de datos ni otro tipo de retraso.
 - **30%** Son **"Branches"** de los cuales
 - 2/3** partes salta (es decir, es verdadero, o tomado)
 - **10%** Son **sw's** (guarda el valor de un registro en memoria -caché de datos-) para los que
 - la mitad**, tiene conflicto de datos con un lw inmediato anterior para el valor que guarda en memoria
 - la otra mitad**, tiene conflicto de datos para el valor del registro que sirve para calcular la dirección de memoria, con una operación aritmética inmediata anterior.
 - el restante **20%** tiene un **CPI = 1**
- a. **Calcule el CPI** para este programa. Para cada caso debe hacer una "tablita de tiempos" que justifique el valor que asigna. Además, explique lo que **significa que el CPI** de ese programa tenga ese valor, es decir, explique lo que indica el CPI para ese programa.
- b. Si ahora el procesador y el compilador, trabajaran con la estrategia de **"branch retrasado"**, incluyendo **2 instrucciones buenas** luego del branch ¿qué aceleración lograría este programa? Haga las **tablas de tiempos** requeridas, de manera que se justifique su resultado. **Además**, explique cómo trabaja esta estrategia de branch retrasado.
2. Escriba una secuencia de instrucciones RISC-V, las cuales al correr en el pipeline visto en clase con "Forwarding" obligue a que se realice un forwarding desde el registro **M/WB.AluOutput** al **segundo operando del ALU**. Escriba la tabla de tiempos y señale el forwarding con un flecha.
3. ¿Qué tan importante es la **definición de la arquitectura del conjunto de instrucciones** de un procesador en relación con **su diseño lógico** de un procesador? Indique al menos 3 de las características que se definen como parte de esta arquitectura.
4. Supongamos que se desea evaluar, **solo en términos de los datos que se presentan a continuación**, estos 3 procesadores:

	A	B	C
Usa reglas de alineación para direcciones de memoria	Sí	No	Sí
Número de etapas para el pipeline de operaciones con enteros	8	5	20
Velocidad del reloj	2.8 GHz	2 GHz	3 GHz
Trabaja con la estrategia de branch Retrasado	Sí	Sí	Sí
Número de núcleos	4	4	2

Basándose solo en estos datos presentados,

- a. Elija a uno de estos procesadores como **el mejor**. Indique por qué lo escogió comparándolo con los otros dos.
- b. A cuál dejaría de último (**el más malo** entre los 3). Explique por qué
5. Entre las estrategias para reducir los ciclos de retraso por conflictos de control, es decir, por branches, están:
la de hacer "**predicción no tomado**" de manera estática y la de **branch retrasado**.
- Suponiendo que para el branch retrasado se incluyen **2 instrucciones buenas** por parte del compilador en un procesador RISC-V, ¿cuál de estas dos estrategias resulta mejor? Explique por qué. Ponga ejemplos.
6. Como parte de la **definición de la arquitectura del conjunto de instrucciones** de un procesador, está la definición de la **codificación de las instrucciones (formatos de codificación)**. ¿Qué tan importante es esto y cómo afecta la manera en la que el procesador se desempeñe?

7. Supongamos que se desea evaluar, **solo en términos de los datos que se presentan a continuación**, estas 3 computadoras: la **A**, la **B** y la **C**.

	A	B	C
Núm de procesadores	2	4	2
Núm de núcleos por procesador	2	4	8
Núm de hilos por núcleo	4	1	1
Núm total de cachés de instrucciones Nivel 1	4	8	16
Número total de cachés de Datos Nivel 1	2	8	16

Basándose solo en estos datos presentados,

- a. Elija **la mejor**, e indique por qué la escogió comparándola con las otras dos.
- b. A cuál dejaría de última (**la más mala** entre las 3). Explique por qué
8. Un programa en lenguaje de alto nivel contiene el **siguiente ciclo: (A y B dos vectores enteros de longitud 32, en donde A se encuentra en memoria a partir del byte 0 y B a partir del byte 128)**

```
for (int i = 0; i < 32, i++) {
    A[i] = A[i] + B[31-i];
}
```

Un compilador para RISC-V produjo el siguiente código para dicho for. Se sabe **que x1** apunta al primer elemento de A (dirección 0) y **x2** al último de B (dirección 252). El valor de **x3 es 32**, es decir, la dimensión de A y de B.

Compilador		
Aquí	lw	x15, 0(x1)
	lw	x16, 0(x2)
	add	x15, x15, x16
	sw	x15, 0(x1)
	addi	x1, x1, 4
	addi	x2, x2, -4
	addi	x3, x3, -1
	bnez	x3, Aquí

- a. Ese fragmento de código correrá en un procesador RISC-V que tiene una caché de datos nivel 1 con capacidad para 8 bloques de 4 palabras cada una, **caché Write Back para aciertos de escritura y Write Allocate para fallos de escritura**, de Mapeo directo para asignación de bloques.

Calcule el número de fallos de caché y de palabras que se envían a escribir a memoria. Explique sus cálculos.

- b. Calcule lo mismo que en a) pero para cuando la caché es Write Through para aciertos de escritura y no Write Allocate para fallos.

CACHE																																
0				1				2				3				4				5				6				7				

MEMORIA PRINCIPAL																																
0				1				2				3				4				5				6				7				
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100	104	108	112	116	120	124	
A[i]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

8				9				10				11				12				13				14				15				
128	132	136	140	144	148	152	156	160	164	168	172	176	180	184	188	192	196	200	204	208	212	216	220	224	228	232	236	240	244	248	252	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	B[31-i]

Resp. para a) dan 16 fallos y se escriben 16 palabras a memoria. La razón es que A va llenando la caché a partir de la posición 0 con el bloque 0, y B comienza llenando caché pero en orden inverso ya que el primer bloque que se lee de B es el 15, el cual se sube a la caché en la posición 7. Cuando se han subido y modificado los primeros 4 bloques de A (0, 1, 2 y 3) que quedan en las posiciones 0, 1, 2 y 3 de caché, también se han subido 4 bloques de B (el 15, 14, 13 y 12, los cuales se almacenaron en las posiciones 7, 6, 5 y 4 de caché. Hasta ese momento el total de fallos es 8 (4 por parte de A y 4 de B) y no se ha escrito nada a memoria pues los 4 bloques modificados de A siguen en Caché. Cuando se lean los siguientes 4 bloques de A, estos van a quedar sobre los 4 bloques de B que ya estaban en caché, pero ninguno de ellos modificado, por lo que solo se contabilizan 4 fallos más pero al irse leyendo los siguientes 4 bloques de B en orden inverso, el 11, 10, 9 y 8 (que provocan 4 fallos más) todos van a colocarse en caché sobre bloques modificados de A, por lo que las 4 palabras de cada uno de esos 4 bloques de A se escribirán a memoria, en total **16 palabras a memoria**. Los otros 4 bloques de A quedan en caché modificados, sin que se tengan que escribir por "culpa de este fragmento de código".

Resp. para b) para cuando la caché es Write Through, no write Allocat, la cantidad de fallos no cambia, pues igual hay que leer 8 bloques de A y 8 de B (**16 fallos en total**) memoria. Pero en cuanto a las palabras escritas a memoria ocurre que por ser write through cada vez que hay un sw (el cual siempre da acierto), se escribe la palabra en caché y se envía también a escribir a memoria, por lo que, el total de **palabras escritas a mem es igual al número de elementos de A, o sea 32**.

9. Suponga que se tiene un procesador de **un núcleo** que tiene solo una **caché de datos Wth, NO W.ALL, Mapeo directo** para asignación de bloque, **bloques de 2 palabras** y con capacidad para **8 bloques**.

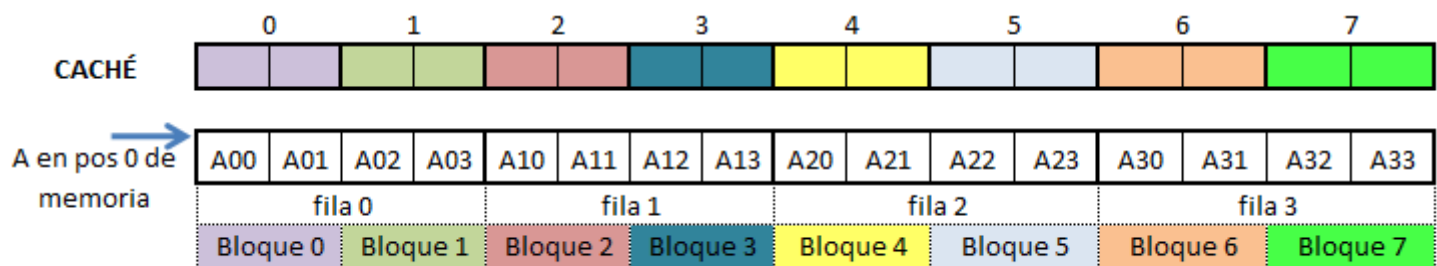
En memoria está la matriz de enteros **A** almacenada por filas a partir de la posición 0 de memoria. Consta de **4 filas por 4 columnas**.

Suponga que ahí corre este fragmento de código

```
for i=0, i < 4. i++
    for j= 0, j<4, j++ {
        x += A[ i ][ j ];
        A[ i ][ j ] ++;
    }
```

Calcule el número de fallos de caché y de palabras escritas a memoria que se dan durante la ejecución de dicho código. Explique sus cálculos

A tiene 16 entradas de 1 pal cada una, por lo que está compuesta por 8 bloques, del bl 0 al 7. Al ir leyendo cada dato de una columna par va a dar fallo, para los datos de columnas impares ya no da fallo. **En total 8 fallos**. Cada una de las modificaciones de las entradas de A al sumarles 1 quedan en caché y también se envían a escribir a memoria principal, por lo que hay **16 palabras escritas a memoria**.



10. Suponga que se tiene un procesador de **un núcleo** que tiene solo una **caché de datos WB, W.ALL, Mapeo directo** para asignación de bloque, **bloques de 2 palabras** y con capacidad para **8 bloques**.

En memoria está la matriz de enteros **A** almacenada por filas a partir de la posición 0 de memoria. Consta de **4 filas por 4 columnas**. También el vector de enteros **V** con **8 elementos**, almacenado a partir del byte **80** de memoria.

Suponga que ahí corre este fragmento de código

```

for i=0, i < 4, i++
  for j= 0, j<4, j++ {
    x += A[i][j];
    A[i][j] ++;
  }

for i=0, i < 8, i= i+2
  y += V[i+1];

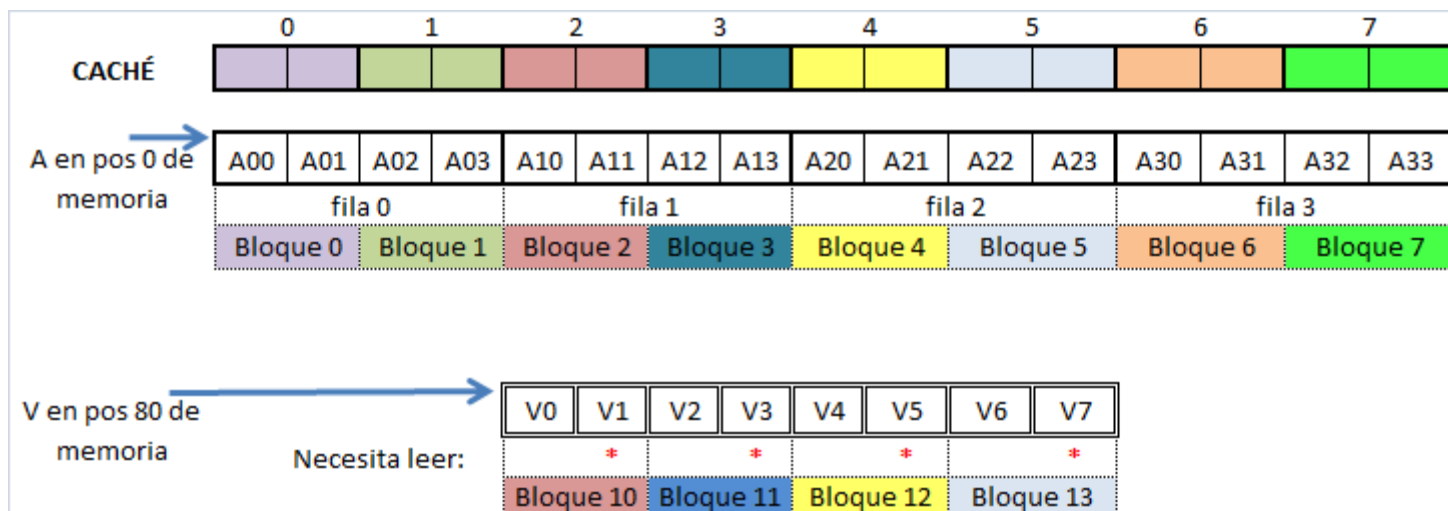
```

Calcule el número de fallos de caché y de palabras escritas a memoria que se dan durante la ejecución de dicho código. Explique sus cálculos

R/ A tiene 16 entradas de 1 pal cada una, por lo que está compuesta por 8 bloques, del bl 0 al 7, al ir leyendo cada dato de una columna par va a dar fallo, para los datos de columnas impares ya no da fallo. **En total 8 fallos**, en ningún caso el bloque del fallo debe colocarse en una posición de caché que ya contiene un bloque de A, por lo que no se reemplazan bloques modificados. Nótese que cada una de las modificaciones de las entradas de A al sumarles 1 quedan en caché, y el bloque queda **modificado**, por lo que **no se hacen escrituras a memoria para la parte de A**. Al final de las lecturas y escrituras A queda completa en la caché.

V tiene solo 8 elementos y de ellos solo se necesita leer V[1] V[3] V[5]y V[7], pero igual deben leerse los 4 bloques que componen V ya que cada uno de estos valores está en un bloque distinto (bloques 10, 11, 12 y 13 de mem respectivamente). Por esto se darán **4 fallos** de caché, para las posiciones 2, 3, 4 y 5 de la caché, en las que se encuentran los bloques 2, 3, 4 y 5 correspondientes a la matriz A y que **están modificados**, por lo que se debe escribir a memoria cada uno antes de subir el bloque correspondiente de V, así tendremos **8 palabras escritas a memoria**


En total se tendrán 12 fallos de caché y se escribirán 8 palabras a memoria



11. Indique en cuál o cuáles estrategias de asignación de bloques en caché (formas de decidir en qué posición de la caché se va a almacenar o copiar un bloque) tiene utilidad aplicar también estrategias de reemplazo de bloques (**LRU, FIFO, Random**) ¿por qué? ¿qué utilidad tiene usarlas?
12. Si mi programa (con mismas condiciones iniciales), al pasar de correr del procesador **A** al procesador **B** logra una aceleración igual a **4**, pero al pasar de correr del procesador **A** al procesador **C** logra una aceleración igual a **3**, ¿Cuál es la aceleración que lograría si se pasa de correr de el procesador **C** al procesador **B**? **Haga sus cálculos**
13. Suponga un procesador *pipelineado* en el que el pipeline por el cual pasan los saltos condicionales o *branches* durante su ejecución tiene **8 etapas**. Los "branches" **modifican el PC durante la etapa 6**, es decir, para la etapa 7 del branch ya se sabe cuál instrucción sigue. Si se quisiera utilizar la estrategia de **branch retrasado** ¿cuál es el máximo de instrucciones buenas que podría colocar el compilador luego de la instrucción **branch**? **Explique por qué.**
14. Con base en las siguientes imágenes y sus recuadros, obtenidos con el programa CPU-Z, compare el procesador **Core i-9 de Intel** (de **3.3 GHz**) el cual salió al mercado en Junio de 2017 y el procesador **Ryzen 7 de AMD** (**3.6 GHz**) que salió en marzo del 2017 también, ambos para computadoras personales. Tome en cuenta todas las propiedades posibles de acuerdo con lo que se vió en el curso. **No dé una respuesta muy extensa.**

Para ayudar a entender:

- En donde se indica que una caché es "x-way" lo que significa es que su estrategia para asignación de bloques en la caché es **asociativa por conjuntos de "x" vías**, es decir, que cada conjunto es de **x** bloques.
- Cuando se indica **Core Speed**, esa es la velocidad actual del reloj del core indicado, que se calcula como **Bus Speed x Multiplier**. Ese bus se refiere en realidad al modo de conexión entre CPU y el controlador de la memoria principal, y trabaja a cierta velocidad. Ese multiplicador puede variar durante la operación del procesador.
- Cuando **no se indica si una caché es de datos o instrucciones**, esto significa que es "**unificada**", es decir, que es tanto de datos como de instrucciones.
- Junto a cada caché descrita se pone un valor entero multiplicando su tamaño, ese valor indica el **número de esas cachés que tiene el procesador**.

CPU	Caches	Mainboard	Memory	SPD	Graphics	Bench	About
Processor							
Name	Intel Core i9 7900X						
Code Name	Skylake-X	Max TDP	140.0 W				
Package	Socket 2066 LGA						
Technology	14 nm	Core VID	1.328 V				
Specification	Intel® Core™ i7-7900X CPU @ 3.30GHz (ES)						
Family	6	Model	5	Stepping	4		
Ext. Family	6	Ext. Model	55	Revision	H0		
Instructions	MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, AVX512F, FMA3, TSX						
Clocks (Core #0)				Cache			
Core Speed	4500.06 MHz			L1 Data	8 x 32 KBytes	8-way	
Multiplier	x 45.0 (12 - 43)			L1 Inst.	8 x 32 KBytes	8-way	
Bus Speed	100.00 MHz			Level 2	8 x 1 MBytes	16-way	
Rated FSB				Level 3	13.75 MBytes	11-way	
Selection				Cores			
Socket #1				8			
				Threads			
				16			

CPU | Caches | Mainboard | Memory | SPD | Graphics | Bench | About

Processor

Name

AMD Ryzen 7 1800X

Code Name

Summit Ridge

Max TDP

95.0 W

Package


Socket AM4 (1331)

Technology

14 nm

Core Voltage

0.960 V



Specification

AMD Ryzen 7 1800X Eight-Core Processor

Family

F

Model

1

Stepping

1

Ext. Family

17

Ext. Model

1

Revision

ZP-B1

Instructions

MMX(+), SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, SSE4A, x86-64, AMD-V, AES, AVX, AVX2, FMA3, SHA

Clocks (Core #0)

Core Speed

4103.81 MHz

Multiplier

x 38.0

Bus Speed

107.99 MHz

Rated FSB

Cache

L1 Data

8 x 32 KBytes

8-way

L1 Inst.

8 x 64 KBytes

4-way

Level 2

8 x 512 KBytes

8-way

Level 3

2 x 8 MBytes

16-way

Selection

Socket #1

Cores

8

Threads

16

15. Con base en los recuadros de la siguiente imagen, obtenida con el programa CPU-Z y en la que se describen algunas características del procesador para computadoras portátiles **core-i5 4300M de Intel**, que salió al mercado en el 2014b **a) dibuje un diagrama** (esquema simple) que describa este procesador incluyendo sus jerarquías de memoria. **b)** Además describa con sus palabras, **todas las propiedades o características de este procesador y qué implican en cuanto a su desempeño** . La idea es que use lo que se vió en el curso. Sea breve.

Para ayudar a entender:

- En donde se indica que una caché es "x-way" en la fig de la izquierda, o "x-way set associative" en la imagen de la derecha, lo que significa es que su estrategia para asignación de bloques en la caché es **"asociativa por conjuntos de x vías"**, es decir, que cada conjunto es de x bloques.
- Cuando se indica **Core Speed**, esa es la velocidad actual del reloj del core indicado, que se calcula como **Bus Speed x Multiplier**. Ese bus se refiere en realidad al modo de conexión entre CPU y el controlador de la memoria principal, y trabaja a cierta velocidad. Ese multiplicador puede variar durante la operación de dicho procesador.
- Cuando **no se indica si una caché es de datos o instrucciones, esto significa que es "unificada"**, es decir, que es tanto de datos como de instrucciones.
- Junto a cada caché descrita se pone un valor entero multiplicando su tamaño, ese valor indica el **número de esas cachés que tiene el procesador**.
- "Line Size" es el **tamaño del bloque en la caché**

CPU | Caches | Mainboard | Memory | SPD | Graphics | Bench | About

Processor

Name: Intel Core i5 4300M

Code Name: Haswell Max TDP: 37.0 W

Package: Socket 947 rPGA

Technology: 22 nm Core Voltage: 0.673 V

Specification: Intel® Core™ i5-4300M CPU @ 2.60GHz

Family: 6 Model: C Stepping: 3

Ext. Family: 6 Ext. Model: 3C Revision: C0

Instructions: MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, FMA3, TSX

Clocks (Core #0)

Core Speed: 798.10 MHz

Multiplier: x 8.0 (8 - 33)

Bus Speed: 99.76 MHz

Rated FSB:

Cache

Cache	Size	Way
L1 Data	2 x 32 KBytes	8-way
L1 Inst.	2 x 32 KBytes	8-way
Level 2	2 x 256 KBytes	8-way
Level 3	3 MBytes	12-way

Selection: Socket #1

Cores: 2 Threads: 4

CPU | Caches | Mainboard | Memory | SPD | Graphics | Bench

L1 D-Cache

Size: 32 KBytes x 2

Descriptor: 8-way set associative, 64-byte line size

L1 I-Cache

Size: 32 KBytes x 2

Descriptor: 8-way set associative, 64-byte line size

L2 Cache

Size: 256 KBytes x 2

Descriptor: 8-way set associative, 64-byte line size

L3 Cache

Size: 3 MBytes

Descriptor: 12-way set associative, 64-byte line size

Size:

Descriptor:

Speed:

16. Relacionado con el gráfico de la FIG 1.1 del libro, y de la presentación vista en la clase anterior,
a) Explique qué causó el aumento tan pronunciado de la curva de este gráfico entre los años 1986 y 2003.

R/ Este aumento en rendimiento se debió principalmente a la introducción de la ejecución de las instrucciones del conjunto de instrucciones en paralelo (pipelining), así como de la continua mejora de esta técnica. También contribuyó a esto la miniaturización de los transistores que permitió el aumento de funcionalidad por unidad de superficie en el procesador (Esto último no lo mencionamos en clase, por lo que no espero que lo hayan incluido en su respuesta)

- b) Suponiendo que para calcular estas medidas de desempeño del gráfico se utilizó el programa de prueba X, calcule la aceleración que lograría el programa X al pasarse de ejecutar de **la HP 9000/750 de 66 MHz** (la cual salió al mercado aprox entre 1990 y 1992) **a la IBM Power4, de 1.3 GHz** (la cual salió aprox entre el 2000 y el 2002)

R/

-El programa X, cuando se pasó a correr **de la VAX a la HP 9000/750 de 66 MHz** obtuvo una aceleración igual a **51** o sea que:

$$\frac{\text{tiempo ejecución programa X en la VAX}}{\text{tiempo de ejecución del programa X en la HP 9000/750}} = \frac{T_{ej} X VAX}{T_{ej} X HP} = 51 \Rightarrow T_{ej} X HP = \frac{T_{ej} X VAX}{51}$$

- El programa X, cuando se pasó a correr **de la VAX a la IBM Power4 de 1.3 GHz** obtuvo una aceleración de **3016** o sea que:

$$\frac{\text{tiempo ejecución programa X en la VAX}}{\text{tiempo de ejecución del programa X en la IBM Power4}} = \frac{T_{ej} X VAX}{T_{ej} X IBM} = 3016 \Rightarrow T_{ej} X IBM = \frac{T_{ej} X VAX}{3016}$$

Como lo que se desea calcular es la aceleración que lograría el programa X al pasarse de correr de la HP 9000/750 a la IBM Power4 lo que se debe calcular es:

$$\frac{T_{ej} X HP 9000/750}{T_{ej} X IBM Power4}$$

Estos tiempos los obtuvimos antes en términos de el tiempo de ejecución del programa X en la VAX. Por lo tanto:

$$\frac{T_{ej} X HP}{T_{ej} X IBM} = \frac{T_{ej} X VAX / 51}{T_{ej} X VAX / 3016} = \frac{3016/51}{1} = \mathbf{59.1373}$$

La aceleración que lograría el programa X al pasarse de correr de la HP 9000/750 de 66 MHz a la IBM Power4 de 1.3 GHz es igual a **59.1373**. Es decir, mientras el programa X corre 1 vez en la HP correría 59.1373 veces en la IBM

17. ¿qué es el **conjunto de instrucciones** de una computadora?

El conjunto de instrucciones se compone de todas las instrucciones que el procesador de esa computadora puede ejecutar. Es decir, son aquellas instrucciones que se ejecutan directamente por el *hardware*.

18. ¿Qué es la **arquitectura del conjunto de instrucciones** de una computadora?

La arquitectura del conjunto de instrucciones está definida por:

“ la manera de almacenar los operandos para las operaciones que ejecuta el procesador (RR-en registros, RM-en registro y memoria, en una pila de memoria, un operando en el registro llamado "acumulador" y otro operando en la memoria) Si es RR es RISC

- “ las distintas formas de direccionar la memoria (por byte, por palabra, si debe estar alineada, el tamaño dirección)
- “ de cuántas maneras se puede indicar al procesador cómo calcular la dirección de memoria para acceder a un dato (lectura o escritura)
- “ los distintos tipos de datos que maneja el procesador, así como los diferentes tamaños permitidos para estos.
- “ Qué operaciones aritméticas, lógicas, de transferencia de datos, de control se pueden ejecutar.
- “ La manera de codificar las instrucciones de manera que el procesador las pueda comprender. Es decir, cómo es el lenguaje de máquina.

19. Relacionado con **procesadores RISC y CISC**

a) Explique que es una computadora CISC y qué es una RISC.

Una computadora CISC (con procesador CISC) es aquella cuyo conjunto de instrucciones es complejo. Es decir, que hay muchos tipos de operaciones, de direccionamientos de memoria, de maneras de almacenar los operandos. En las máquinas CISC los formatos de codificación de las instrucciones suelen ser de longitud variable. Además la ejecución de las instrucciones se hace por medio de la ejecución de un microprograma almacenado en una memoria ROM del procesador.

Una computadora RISC es aquella cuyo conjunto de instrucciones es reducido, simple. Por ello hay pocas operaciones, pocos y simples modos de direccionar memoria, los operandos siempre deben estar almacenados en registros propios del procesador. Los formatos de codificación son pocos y usualmente de longitud fija. El control no es microprogramado, sino que está "alambrado".

b) ¿Cuál es la ventaja de cada una?

Para una computadora con procesador CISC podemos decir que su principal ventaja es que programar en ensamblador (o inclusive en lenguaje de máquina) es simple, pues se cuenta con muchas instrucciones. Otra ventaja es que al ser de control microprogramado, es posible cambiar la manera de ejecutarse de cualquier instrucción, tan solo modificando el microprograma de esta.

Para una computadora RISC se tienen muchas ventajas, entre ellas, que al ser pipelineadas, es decir, al soportar el paralelismo en el nivel de instrucciones, los programas, aunque tengan un código de mayor tamaño que el que tendrían para una máquina CISC, se ejecutan mucho más rápido que en una CISC. Otra ventaja es que al ser el hardware del procesador, diseñado para ejecutar solo un grupo reducido de instrucciones, ese hardware puede hacerse de mucha mejor calidad que aquel que sirve para realizar muchas instrucciones distintas (aunque sean microprogramados).

c) ¿cuál es la desventaja?

En realidad las ventajas de una son las desventajas de la otra.

Pero bueno, lo diré, porque igual se los pregunté sin pensar en eso que les acabo de indicar.

Desventajas de una CISC: lentitud, no puede ejecutar instrucciones en paralelo, gastan mucha energía.

Desventajas de una RISC: códigos muy grandes (en lenguaje de máquina), lo cual hace tedioso programar en ensamblador. Si se quiere modificar la manera de ejecutarse una instrucción... pues se debe tirar el procesador y hacer otro... pues a diferencia de las CISC, que son microprogramadas, el control de las RISC es alambrado.

d) ¿Por qué RISC favorece a que se pueda implementar la ejecución de instrucciones de máquina en paralelo (pipelining)?

Si un procesador ejecuta solo un grupo pequeño de instrucciones simples, al dividirse en subetapas el proceso de ejecución de cada instrucción, resulta que muchas de esas subetapas son compartidas por las demás instrucciones.

Por ejemplo:

- todas las instrucciones deben ser leídas de la caché de instrucciones y copiadas en un registro.
- Todas deben decodificarse (averiguar qué hace esa instrucción)
- Casi todas ocupan buscar operandos para realizar su operación (leer valores de los registros del procesador)
- Casi todas las instrucciones usan un ALU para hacer una operación aritmética,
- Casi todas guardan su resultado en un registro.
- Algunas (loads y stores) ocupan trabajar con un dato de la caché de datos.

Entonces si se crea hardware específico para hacer cada una de estas subetapas, es decir:

- hardware que se encargue solo de leer una instrucción de la caché de instrucciones,
- hardware que solo decodifique instrucciones,
- hardware que solo lea de registros los operandos de una instrucción,
- otro hardware para guardar un valor en registros, y
- otro para usar caché de datos,

ocurrirá entonces que las instrucciones pueden ir usando en sus subetapas de ejecución el hardware correspondiente para esta, y una vez que lo desocupa y pasa a la siguiente subetapa de ejecución, libera ese hardware para que la siguiente instrucción lo utilice. De esa manera se pueden tener varias instrucciones ejecutándose al mismo tiempo, solo que cada una usando hardware diferente, es decir, cada una realizando una subetapa diferente.

Eso es ejecutar instrucciones en paralelo o sea "pipelinear" instrucciones. Si el conjunto de instrucciones a ejecutar fuera muy complejo y extenso, es muy difícil que la ejecución de las instrucciones se pueda dividir en etapas similares para la mayoría de las instrucciones, y de manera que no deban usar el mismo hardware. Por ejemplo si una instrucción lee sus operandos de memoria, ocurriría que una subetapa es ir a caché de datos para leer un operando, pero otra subetapa es volver a pedir a caché otro dato (el 2do operando), otra subetapa es usar el alu, y otra guardar el resultado en caché de datos. Nótese que usaría el mismo hardware en 3 subetapas de ejecución, lo cual ya evitaría el pipeline.

e) ¿Cómo funcionan los procesadores híbridos (CISC/RISC)?

Tienen una especie de "front-end" para recibir (ir leyendo) instrucciones CISC. Luego un hardware específico convierte cada instrucción CISC en varias instrucciones RISC, las cuales se van enviando a colas de ejecución, dependiendo del tipo de instrucción, para que comiencen a ejecutarse en "pipeline".

20. Señale ventajas y desventajas de los procesadores híbridos

Ventajas: facilitan el trabajo a los programadores en ensamblador, así como el de los compiladores, ya que al procesador se le envían instrucciones CISC para ejecución. A la vez son rápidos en la ejecución de programas, pues las instrucciones CISC se pasan internamente a instrucciones RISC, y estas son las que se ejecutan, por lo que tienen implementados pipelines.

Desventajas:

Son complejos de diseñar y gastan mucha energía pues realizan la conversión al RISC y además ejecutan instrucciones RISC al mismo tiempo.

21. En el procesador Ceónico cuya velocidad es de 2.85 GHz corrió el programa X y tardó ejecutándose 7000 ciclos de reloj. En el procesador UltraCe, de 2.4 GHz el mismo programa X, y con los mismos datos iniciales, tardó ejecutándose 5000 ciclos de reloj (en ambos casos solo se considera el tiempo de uso de CPU)

a) Calcule cuantos ns (nanosegundos) tarda ejecutándose el programa X en cada una de las computadoras

SOLUCIÓN:

Se debe calcular el tiempo de ejecución (tiempo de CPU) del programa para cada uno de los procesadores:

T. Ejec. en procesador Ceónico = 7000 ciclos * LCRc

T. Ejec. en procesador UltraCe = 5000 ciclos * LCRu

Para Calcular las longitudes del ciclo de reloj en cada procesador se utiliza su velocidad, la cual indica el número de ciclos de reloj en un segundo:

Ceónico:

Como en un segundo se dan 2.85×10^9 ciclos de reloj, cada ciclo de reloj mide

$1/(2.85 \times 10^9)$ de segundo, es decir $LCRc = 1/(2.85 \times 10^9) = 1/2.85 \times 1/10^9$ segundos = $1/2.85$ ns = 0.3509 ns (ó 20/57 ns)

UltraCe

Por un razonamiento similar al del Ceónico, $LCRu = 1/2.4$ ns = 0.4167 ns (ó 5/12 ns)

Sustituyendo:

T. Ejec. en procesador Ceónico = 7000 ciclos * LCRc = 7000 ciclos * 0.3509 ns = **2456.1404 ns**

T. Ejec. en procesador UltraCe = 5000 ciclos * LCRu = 5000 ciclos * 0.4167 ns = **2083.3333 ns**

RESPUESTA: A pesar de que el Ceónico es más rápido que el UltraCe, **el programa tarda ejecutándose más tiempo en el procesador Ceónico.**

b) Calcule la aceleración que logra el programa al pasar de la computadora en la que tarda más a la computadora en la que se ejecuta más rápido. Escriba y documente sus cálculos.

La aceleración que logra al pasar de correr del Ceónico al UltraCe =

= tiempo de ejecución del programa en el Ceónico / tiempo de ejecución del programa en el UltraCe = $2456.1404 / 2083.3333 = 1.1789$

22. Cuando se define la arquitectura del conjunto de instrucciones de una computadora (ISA), uno de los aspectos a definir es la codificación de las instrucciones.

a. ¿qué es la codificación de las instrucciones?

b. ¿por qué la definición de la codificación de las instrucciones es muy importante para la forma en la que trabajará la computadora que implemente este conjunto de instrucciones?

c. ¿de qué longitud son los formatos de codificación para un procesador RISC-V? o sea, de qué tamaño quedan las instrucciones ya codificadas.

23. En un procesador RISC, en el que obviamente se implementa el paralelismo de instrucciones
- Explique de qué manera es que se logra paralelizar la ejecución de instrucciones
 - Si los procesadores RISC, por su naturaleza, ejecutan solo un conjunto pequeño de instrucciones, ¿por qué es que son más rápidos que un procesador CISC?
 - Cuando se dice que un procesador tiene definido un "pipeline" para la ejecución de cierto tipo de instrucciones ¿qué significa esto?
24. En relación con el pipeline para enteros de RISC-V visto
- ¿cuál es la utilidad de los juegos de registros entre etapas (IF/ID, ID/EX, EX/M y M/WB)? (No se pide explicar para que sirve cada uno, sino en general cual es la utilidad, o para qué se necesitan esos juegos de registros entre etapas del pipeline?)
 - Suponiendo que la instrucción que está en ID es un tipo de instrucción que ocupa un inmediato (constante) ¿por qué al inmediato que forma parte del IF/ID.ID se le hace extensión de signo de manera que ese valor se represente con 32 ó 64 bits dependiendo de si la computadora es de 32 o de 64 bits y luego se copia en el registro ID/EX.M?
 - Cuantos ciclos de reloj, cómo mínimo, tarda ejecutándose una instrucción de enteros que "pasa" por este pipeline? ¿Por qué?
25. Control alambrado-control microprogramado
- ¿Cuál es la diferencia entre un procesador de control microprogramado y uno de control alambrado? (como parte de la respuesta, describa cómo trabaja el control alambrado)
 - ¿qué tipo de control tienen los procesadores híbridos? (entre control alambrado y control microprogramado)
26. a) Explique qué son conflictos de datos cuando se ejecutan instrucciones en paralelo (pipeline), y en qué consiste la estrategia de "forwarding".
- b) ¿Siempre es posible resolver los conflictos de datos utilizando forwarding? ¿Por qué?

No, no siempre el dato que se necesita está en registros intermedios del pipeline, para ser enviados a la instrucción que lo ocupa.

ej: Note que acá no es posible enviar a Mem el valor de x5 que ocupa el sw y que el lw obtuvo al final de su etapa de Mem

lw x5 , 100(x0)	IF	ID	EX	M	WB	
lw x9, 20(x7)		IF	ID	EX	M	WB
sw x5 , 80 (x2)			IF	ID	EX	¿M?

Por lo que es necesario que el sw detenga en ID hasta que el lw x5 haga WB:

lw x5 , 100(x0)	IF	ID	EX	M	WB			
lw x9, 20(x7)		IF	ID	EX	M	WB		
sw x5 , 80 (x2)			IF	ID	ID	EX	M	WB

27. Entre las estrategias para reducir los ciclos de retraso por conflictos de control, es decir, por branches, están: la de hacer "**predicción no tomado**" de manera estática y la de **branch retrasado**. Suponiendo que para el branch retrasado se incluyen **2 instrucciones buenas** por parte del compilador en un procesador RISC-V,

- a) Explique en qué consiste la estrategia de "**branch retrasado**" (**use sus palabras**)
- b) ¿cuál de esas dos estrategias resulta mejor? Explique por qué. Ponga ejemplos.
Si siempre se logra encontrar 2 instrucciones buenas como se indica en el enunciado, entonces el retraso por cada branch se reduce a 0, y la estrategia de branch retrasado es mejor, pues la de predicción no tomado, solo evita el retraso cuando el branch es no tomado.

28. Amalia **diseña compiladores** para el procesador X y Marino **hace programas en lenguajes de alto nivel** que correrían en una máquina con el procesador X. Entre ellos dos, ¿para quién es **indispensable** saber si en ese procesador X se trabaja la **estrategia de branch retrasado**? ¿por qué?

29. **a) Escriba la tabla de tiempos** para el siguiente código RISC-V para enteros, el cual se ejecuta en un procesador con igual *pipeline* al visto en clase, solución de branches en EX, **con predicción NO TOMADO** para branches y con **forwarding** para conflictos de datos. Utilice flechas que muestren con total claridad de donde en donde se realizan los "forwardings". **Calcule el CPI** e indique lo que significa

b) Suponiendo que el primer lw está en la dir de memoria 100, ¿cuánto vale el NPC al final del ciclo 5?

Valores iniciales:

M[20] = 17

M[160] = 80

M[120] = 4

x3 = 20

x2 = -15

Código RISC-V:

lw x5, 160(x0)

lw x8, 40(x5)

sub x4, x5, x3

bnez x8, etiq

add x4, x4, x2

lw x8, 20(x0)

etiq sw x4, 200(x0)

addi x8, x4, 0

div x6, x4, x8

sw x6, 40(x0)

Código

RISC-V:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
lw x5, 160(x0)	IF	ID	EX	M	WB													X5=80
lw x8, 40(x5)		IF	ID	EX	EX	M	WB											X8=4
sub x4, x5, x3			IF	ID	ID	EX	M	WB										X4=60
bnez x8, etiq				IF	IF	ID	EX	M	WB									SALTA Y NPC= 116
add x4, x4, x2						IF	ID											
lw x8, 20(x0)							IF											
etiq: sw x4, 200(x0)								IF	ID	EX	M	WB						M[200]=60
addi x8, x4, 0									IF	ID	EX	M	WB					x8=60
div x6, x4, x8										IF	ID	EX	M	WB				x6=1
sw x6, 40(x0)											IF	ID	EX	M	WB			M[40]=1

CPI= $(1+2+1+1+3+1+1+1)/8 = 11/8 = 1.375$ Significa que en promedio, cada instrucción finaliza 1.375 ciclos después de que finaliza la anterior.

30. Para el siguiente fragmento de código RISC-V,

a) haga su tabla de tiempos,:

Suponiendo que estos son los valores iniciales necesarios:

x8 = 40

M[80] = 30

x9 = 5

x10 = 3

x20 = -4

x15=2

x13= 9

	1	2	3	4	5	6	7	8	9	10	11	
lw x5, 40(x8)	IF	ID	EX	M	WB							x5=30
add x6, x9, x10		IF	ID	EX	M	WB						x6=8
addi x7, x20, 7			IF	ID	EX	M	WB					x7=3
sw x8, 130(x5)				IF	ID	EX	M	WB				M[160]=40
sub x11, x15, x13					IF	ID	EX	M	WB			x11=-7

b) Indique:

-Valor del ALUOutput en los registros EX/M al final del ciclo de reloj 3

Vale 80

-Valor del LMD en M/WB al final del ciclo 4

30

-Valor en posición de memoria 160 al final del ciclo 7

40

-Valor de ID/EX.B al final del ciclo 6

9

c) Explique qué está haciendo cada una de las instrucciones que están en el pipeline, durante el ciclo 6 (no solo en qué etapa están, sino también qué hacen en ésta)

31. Para el siguiente código RISC-V, escriba **la tabla de tiempos**, calcule el **CPI**, indique **"forwardings"** con una flecha e indique **el registro** del cual se necesita su valor. (Para op. de flotantes, recuerde: suma 4 ciclos, multiplicación, 7)

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
fadd	f7, f10, f11	IF	ID	A1	A2	A3	A4	M	W									
fsd	f7, 400(x15)		IF	ID	EX	EX	EX	EX	M	W								f7
fmul	f1, f7, f2			IF	ID	M4	M4	M1	M2	M3	M4	M5	M6	M7	M	W		f7
sub	x16, x16, x15				IF	ID	ID	ID	EX	M	W							
fld	f7, 80(x0)					IF	IF	IF	ID	EX	M	W						
addi	x15, x16, #17								IF	ID	EX	M	W					x16

$$\text{CPI} = (1+1+1+1+1+3)/6 = 8/6 = 4/3$$

32. Asumiendo cachés y bloques del mismo tamaño ¿cual estrategia de asignación de bloque causa una tasa mayor de fallos? ¿por qué? y ¿cuál causaría una menor tasa de fallos? ¿por qué?. Entre estas dos indicadas, ¿cuál es la más lenta y compleja de implementar? Explique

33. a) Explique qué es un núcleo superescalar. b) ¿Por qué un programa no paralelizado podría tener un $CPI < 1$ al correr en un núcleo superescalar? c) ¿qué significa que un núcleo es multihilo? d) ¿qué tipos de multihilo conocimos en el curso? Explique cada uno, e indique cuál de ellos no se podría implementar si el núcleo no es superescalar. ¿por qué?

34. En el pipeline para enteros de la MIPS R4000, la etapa de MEM que "era" muy similar a la etapa MEM de RISC-V, se ha dividido en **3 subetapas pipelineadas: DF-DS-TC**.

- a) Explique qué hace en cada una de ellas un **load**
- b) Explique por qué no tiene sentido pipelinear stores a través de estas 3 subetapas
- c) ¿Qué estrategia de asignación de bloques tiene sentido utilizar en la caché L1 de datos en un núcleo con este pipeline para enteros? ¿por qué?

35. Suponga que en un procesador RISC-V y el cual se detiene con branches hasta conocer el resultado, corre un programa con un **$CPI = 5.2$ tomando en cuenta todos los posibles retrasos** y cuya mezcla de instrucciones contiene un **30% branches**, de los cuales el 80% es tomado. ¿Cuál sería la aceleración de este programa si se recompila con el mismo compilador que se usó antes, pero aplicando ahora la **estrategia de 1 branch retrasado**, con la cual se sabe que van a colocarse en este programa **2 instrucciones buenas** del mismo programa (No NO Ops) inmediatamente luego de cada %Branch+del código? (asuma que se correría en el mismo procesador con la diferencia de que ahora **sí trabaja con branch retrasado**)

Para $CPI = 5.2$ los retrasos por culpa de los branches es: $0.3 * 2 = 0.6$ (tomados y no tomados porque la estrategia no hace diferencia en retraso)

	1	2	3	4	5	6	7	8	
beq xi, x2, Et	IF	ID	EX	M	WB				
INST 1		IF							
Et. Ó INST 1				IF	ID	EX	M	WB	$CPI = 3 \Rightarrow \text{RETRASO} = 2 \text{ CICLOS}$

Entonces CPI sin esos retrasos = $5.2 - 0.6 = 4.6$ (igual se pudo hacer $CPI_n = 5.2 - 0.3 * 3 + 0.3 * 1 = 4.6$)

Con branch retrasado y 2 instrucciones buenas realmente, ahora no habría retraso alguno, por lo que el nuevo CPI se calcula como $5.2 - 0.6 = 4.6$ y la aceleración sería:

$ac = \text{Tiempo Viejo} / \text{Tiempo Nuevo} = IC * CPI_v * LCR / IC * CPI_n * LCR$ (ya que el IC ni la LCR cambian)

$ac = IC * CPI_v * LCR / IC * CPI_n * LCR = 5.2 / 4.6 = 1.1304$

36. Suponga que en un programa RISC-V el 60% de las instrucciones son op.aritméticas pero ninguna tiene conflicto de datos, y el resto (el 40%) son operaciones de memoria, de las cuales el 50% tiene fallo de caché de datos con un **retraso de 4 ciclos**. Suponga que en el fetch de estas instrucciones no se da fallo de caché de intrucciones (siempre se hace **prefetch**). Entonces el CPI de este programa se calcula y da como resultado así:

$$0.6 * 1 + 0.4 * 0.5 * 1 + 0.4 * 0.5 * 5 = 1.8$$

37. Ahora observe esta tabla de tiempos para un programa+RISC-V con flotantes y calcule su CPI (el fld. tuvo fallo de caché de datos, por eso se repite en la etapa Mem) $CPI = 1$

Note que este "programa" cumple con todo lo que se indica para el de la pregunta anterior. ¿Por qué difieren los CPI 's? Explique claramente usando la teoría que conoce.

INST	1	2	3	4	5	6	7	8	9	10	11	12	13	14
fmul f8, f1, f2	If	ID	M1	M2	M3	M4	M5	M6	M7	M	WB			
fld f4, 0(x2)		If	ID	EX	M	M	M	M	M	WB				
fmul f0, f10, f6			If	ID	M1	M2	M3	M4	M5	M6	M7	M	WB	
fadd f2, f11, f9				If	ID	A1	A2	A3	A4	A4	M	WB		
fsd f16, 0(x2)					If	ID	EX	EX	EX	EX	EX	EX	M	WB

Porque la forma como hemos acostumbrado a calcular el CPI es suponiendo que las instrucciones se ejecutan una detrás de otra por el mismo pipeline (como el de enteros de RISC.V), pero cuando agregamos la ejecución de operaciones con flotantes para las que entonces **se tiene paralelismo entre la ejecución de flotantes (entre ellas, y con ejecución de operaciones de enteros y loads y stores-incluyendo sus retrasos por fallo de caché) así como terminaciones fuera de orden**, entonces los retrasos se dan en paralelo con tiempo efectivo de ejecución de otras instrucciones, y eso disminuyen el CPI final.