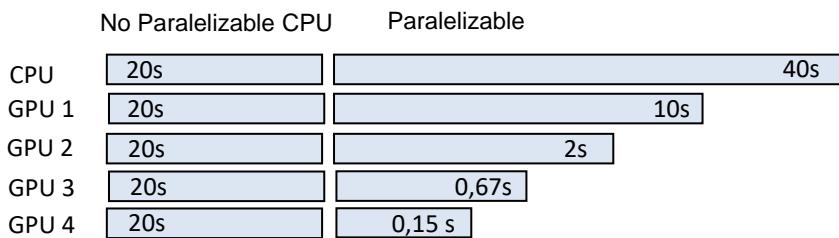


Arquitectura de los Computadores. Primera convocatoria 2018

1. (3 puntos) Sobre rendimiento

La empresa (Wireless Multimedia Sensor Networks) está diseñando microprocesadores especializados (Wireless Multimedia Sensor Processors) (WMSP) y software especializado, para integrarlos en pequeños sensores inteligentes multimedia que implementan protocolos de comunicación y funciones de compresión/descompresión. Estas funciones de compresión/descompresión pueden ser ejecutadas, siguiendo el modelo SIMD, mediante GPUs integradas en los WMSP. Se han diseñado diferentes modelos del sistema WMSP que integran diferentes modelos de GPU (GPU 1,2,3,4) con diferentes prestaciones. Se ha realizado un estudio de las partes del software del sistema WMSP que son paralelizables mediante modelo SIMD y ejecutables en las GPUs integradas y se han realizado pruebas de ejecución en los diferentes sistemas. En la imagen se observan los tiempos de ejecución del sistema completo, incluyendo la parte “no paralelizable” ejecutada en la CPU y la “paralelizable” SIMD ejecutada en las GPUs.

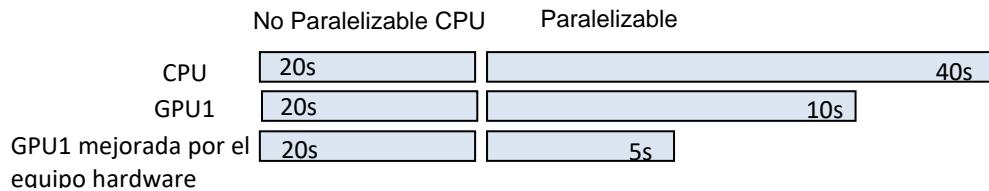


- a) (1 punto) Calcula la aceleración global y la aceleración mejorada respecto a la opción CPU para cada una de las GPUs

	Tiempo no paralelizable	Tiempo paralelizable	Tiempo total = Tiempo no paralelizable + Tiempo paralelizable	A mejorada respecto a la opción CPU de cada una de las GPUs= Tiempo paralelizable CPU / Tiempo paralelizable GPUs	Aceleración global respecto a la opción CPU de cada una de las GPUs= Tiempo total CPU / Tiempo total GPUs
CPU	20,00	40,00	60,00	40/40	1,000
GPU1	20,00	10,00	30,00	40/10	4,000
GPU2	20,00	2,00	22,00	40/2	20,000
GPU3	20,00	0,67	20,67	40/0,67	59,701
GPU4	20,00	0,15	20,15	40/0,15	266,667

Ejercicio inspirado en la página 30 del tema 2 de teoría

- b) (1 punto) El equipo de diseño hardware ha comprobado que es capaz de mejorar la “GPU 1” reduciendo a la mitad los tiempos de GPU (GPU 1 mejorada). Se desea saber si el equipo de desarrollo software puede incrementar el porcentaje de paralelización (fracción mejorada), para mejorar el rendimiento manteniendo la versión actual de la “GPU 1”. ¿Qué incremento en el porcentaje de paralelización se necesitará para obtener la misma ganancia de rendimiento que el equipo hardware?



La aceleración global que puede conseguir el equipo hardware es:

$$A_{g\ hardware} = \frac{20 + 40}{20 + 5} = \frac{60}{25} = 2,4$$

El equipo de desarrollo software utiliza la “GPU1” no mejorada, cuya aceleración mejorada es:

$$A_{mejorada} = \frac{40}{10} = 4$$

El equipo de desarrollo software se plantea la misma aceleración global que el equipo hardware “2,4”, utilizando la “GPU1” no mejorada cuya aceleración mejorada es “4”, incrementando el porcentaje de paralelización o fracción mejorada:

$$A_{g\ hardware} = 2,4 = \frac{1}{(1 - f_{m\ software}) + \frac{f_{m\ software}}{a_m}} = \frac{1}{(1 - f_{m\ software}) + \frac{f_{m\ software}}{4}} \rightarrow f_{m\ software} = 0,78$$

La fracción mejorada inicial (siempre definida sobre la opción peor) es:

$$f_{mejorada\ inicial} = \frac{\text{Tiempo paralelizable CPU}}{\text{Tiempo total CPU}} = \frac{40}{40 + 20} = \frac{40}{60} = 0,67$$

El incremento en el porcentaje de paralelización (fracción mejorada) para obtener la misma ganancia de rendimiento que el equipo hardware (2,4) es:

$$\Delta_{f_m} = |f_{m\ software} - f_{m\ inicial}| = |0,78 - 0,67| = 0,11$$

Ejercicio inspirado en la página 40 del tema 2 de teoría

- c) **(1 punto)** La empresa WMSN está planteando optimizar el rendimiento de la CPU de su WMSP. Se ha realizado un estudio de la mezcla de instrucciones de la CPU ejecutando el software del sistema. El estudio refleja los siguientes resultados de frecuencias de instrucciones y CPIs:

Instrucción	Frecuencia	CPI	CPI nuevo
ALU	45%	1	1
LOAD	12%	2	2
STORE	24%	2	2
SALTO	14%	1	1
PORT	5%	3	4

La CPU utiliza la instrucción PORT para escribir en los puertos de comunicación. Esta instrucción PORT debe ejecutar siempre una instrucción STORE previa para almacenar en memoria los datos que finalmente se escriben al puerto de comunicación. La empresa está pensando en realizar una modificación para que PORT escriba directamente los datos en los puertos de comunicación, sin necesidad de realizar antes una STORE. Esto permitiría transformar parejas de instrucciones STORE/PORT en la nueva instrucción PORT.

$$\begin{array}{l} \text{STORE Reg, Mem} \\ \text{PORT Mem} \end{array} \longrightarrow \text{PORT Reg}$$

Supongamos que esta modificación del repertorio de instrucciones incrementa en 1 el número de ciclos de reloj para la instrucción PORT, pero sin afectar a la duración del ciclo de reloj. **Calcula la aceleración de la versión supuestamente mejorada respecto a la anterior.**

Tiempo de ejecución arquitectura A:

$$T_{ejecución A} = RI_A * CPI_A * clk_A$$

$$CPI_A = 0,45 * 1 + 0,12 * 2 + 0,24 * 2 + 0,14 * 1 + 0,05 * 3 = 1,46$$

$$T_{ejecución A} = RI_A * 1,46 * clk_A$$

Tiempo de ejecución arquitectura mejorada B:

En la arquitectura B, dado que la nueva instrucción PORT escribe directamente los datos en los puertos de comunicación, sin necesidad de realizar antes una STORE, es posible eliminar las instrucciones STORE asociadas a instrucciones PORT. Es decir las parejas STORE; PORT de la arquitectura A se remplazan por PORTn. En esta nueva situación la mezcla de instrucciones y CPIs sería forma:

Instrucción	Frecuencia	CPI	Frecuencia modificada	Frecuencia sobre 100%	CPI
ALU	45%	1	45%	47,4%	1
LOAD	12%	2	12%	12,6%	2
STORE	24%	2	24% - 5% = 19%	20%	2
SALTO	14%	1	14%	14,7%	1
PORT	5%	3	0%	0%	0
PORTn	0%	0	5%	5,3%	4
	100%		100% - 5% = 95%	100%	

$$T_{ejecución B} = RI_B * CPI_B * clk_B$$

$$CPI_B = \frac{0,45 * 1 + 0,12 * 2 + 0,19 * 2 + 0,14 * 1 + 0,05 * 4}{1 - 0,05} = 0,474 * 1 + 0,126 * 2 + 0,2 * 2 + 0,147 * 1 + 0,053 * 4 = 1,48$$

$$clk_B = clk_A$$

$$RI_B = 0,95 * RI_A$$

$$T_{ejecución B} = RI_B * CPI_B * clk_B = 0,95 * RI_A * 1,48 * clk_A = 1,41 * RI_A * clk_A$$

$$A = \frac{T_{ejecución A}}{T_{ejecución B}} = \frac{1,46 * RI_A * clk_A}{1,41 * RI_A * clk_A} = \frac{1,46}{1,41} = 1,0354$$

Ejercicio inspirado en la página 75 del tema 2 de teoría

Arquitectura de los Computadores. Junio 2015

4. (3 puntos) Preguntas sobre segmentación:

- a) **(0,4 puntos)** Considerad un procesador no segmentado con una ruta de datos de 5 etapas de ejecución que funciona a 2GHz, en el que las operaciones ALU y salto requieren cuatro ciclos de reloj y las de memoria cinco. Suponer que las frecuencias relativas para estas operaciones son 45%, 25% y 30% respectivamente. Se quiere segmentar la máquina con 5 etapas y debido al sesgo del reloj y a los registros de segmentación, segmentar el procesador alarga el periodo de reloj en un 10%. El procesador utiliza una cache unificada para datos e instrucciones con un único puerto para el acceso a la memoria lo que provoca un riesgo estructural entre las etapas IF y MEM y consiguiente parada de 1 ciclo de reloj. Suponed que las referencias a datos representan el 30% de las instrucciones ejecutadas y que el CPI ideal del procesador segmentado ignorando el riesgo estructural es 1. ¿Cuál es la ganancia que se puede conseguir con la segmentación?

Solución:

- Sin segmentación:

$$\text{Obtenemos el periodo de reloj: } \text{PeriodoReloj} = \frac{1}{2\text{GHz}} = 0,5\text{ns}$$

Calculamos el CPI medio:

$$CPI = \frac{\sum_{i=1}^n (CPI_i \cdot I_i)}{RI} = (0,45 + 0,25) \times 4 + 0,30 \times 5 = 4,3$$

- Con segmentación:

El periodo de reloj se alarga en un 10% , calculamos el nuevo periodo de reloj:

$$\text{PeriodoReloj} = 0.5\text{ns} + (10\% \times 0.5) = 0.55\text{ns}$$

El 30% de las instrucciones accede a memoria durante la etapa MEM. Cada una de estas instrucciones resulta en un riesgo estructural con una parada de 1 ciclo. El nuevo CPI se puede calcular como:

$$CPI = CPI_{ideal} + Ciclos\ de\ reloj\ detención\ por\ instrucción = 1 + 0.3 \times 1 = 1.3$$

Calculamos ahora la ganancia que se puede conseguir con la segmentación:

$$\text{Ganancia} = \frac{CPI_{sin\ segmentación} \times CLK_{sin\ segmentación}}{CPI_{con\ segmentación} \times CLK_{con\ segmentación}} = \frac{0.5\text{ns} \times 4.3}{0.55 \times 1.3} = 3$$

El procesador segmentado es tres veces más rápido que el no segmentado.

- b) **(0,4 puntos)** Suponer ahora que además se consideran las detenciones por dependencia de datos de 1 ciclo de reloj y que representan el 20% de las instrucciones ejecutadas y las paradas de control de 2 ciclos suponen el 5% de las instrucciones ejecutadas, ¿Cuál es ahora el nuevo CPI?. ¿En qué porcentaje se reduce la ganancia al considerar esta nueva situación?

Solución:

El 20% de las instrucciones ejecutadas tienen además riesgos por dependencia de datos y el 5% riesgos de control con paradas de 2 ciclos de reloj, el nuevo CPI se puede calcular como:

$$CPI = CPI_{ideal} + Ciclos\ de\ reloj\ detención\ por\ instrucción = 1 + 0.3 \times 1 + 0.2 \times 1 + 0.05 \times 2 = 1.6$$

Y la nueva ganancia como:

$$\text{Ganancia} = \frac{CPI_{sin\ segmentación} \times CLK_{sin\ segmentación}}{CPI_{con\ segmentación} \times CLK_{con\ segmentación}} = \frac{0.5\text{ns} \times 4.3}{0.55 \times 1.6} = 2.44$$

La ganancia ha bajado de 3 a 2.44:

$$\left. \begin{array}{l} 3 - 100 \\ 2.44 - x \end{array} \right\} \text{Porcentaje Ganacia Nueva} = \frac{2.44 \times 100}{3} = 81.33\%$$

Nombre: _____

Con esta nueva situación la ganancia ha pasado a ser del 81.33% de la original.

Ha habido una reducción del 18,67% ($Reducción = 100 - 81.33 = 18,67$)

Suponer que el siguiente código MIPS se ejecuta en la máquina segmentada anterior:

```

addi $3, $0, 100
add $4, $0, $0
Loop:   lw $5, 0($1)
        add $4, $4, $5
        lw $6, 0($2)
        sub $4, $4, $6
        addi $1, $1, 4
        addi $2, $2, 4
        addi $3, $3, -1
        bne $3, $0, Loop
    
```

- c) **(1 punto)** Muestra el diagrama de temporización de una iteración del bucle suponiendo que no hay forwarding. Completa la tabla de temporización mostrando todas las todos los ciclos de parada. Suponed que los saltos paran la segmentación durante un ciclo de reloj.

Instrucción	Ciclos de reloj																												
	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			
addi \$3, \$0, 100	IF	ID	EX	M	WB																								
add \$4, \$0, \$0		IF	ID	EX	M	WB																							
lw \$5, 0(\$1)			IF	ID	EX	M	WB																						
add \$4, \$4, \$5				IF	--	--	ID	EX	M	WB																			
lw \$6, 0(\$2)							IF	ID	EX	M	WB																		
sub \$4, \$4, \$6								IF	--	--	ID	EX	M	WB															
addi \$1, \$1, 4											IF	ID	EX	M	WB														
addi \$2, \$2, 4															IF	ID	EX	M	WB										
addi \$3, \$3, -1																				IF	ID	EX	M	WB					
bne \$3, \$0, Loop																					IF	--	--	ID					
lw \$5, 0(\$1)																						IF							
add \$4, \$4, \$5																							IF	--	--	ID	EX	M	WB

- d) **(0,4 puntos)** De acuerdo con el diagrama de temporización del apartado anterior, calcula el número de ciclos de reloj y el CPI medio para ejecutar todas las iteraciones del bucle anterior.

Solución:

Hay 100 iteraciones del bucle (el registro \$3 inicialmente tiene el valor \$3=100).

Cada iteración requiere 15 ciclos como se observa en el cronograma anterior, (8 ciclos para empezar las 8 instrucciones dentro del cuerpo del bucle + 7 paradas).

Además hay 2 ciclos adicionales para comenzar las 2 primeras instrucciones antes del bucle.

Por tanto los ciclos totales necesarios para ejecutar el código serán:

$$Ciclos\ Totales = 100 \times 15 + 2 = 1502\ ciclos \quad (\text{Aproximadamente } 1500\ ciclos)$$

Para el cálculo del número total de instrucciones ejecutadas hay que considerar que dentro del bucle se ejecutan 8 instrucciones, por tanto:

$$Instrucciones\ Ejecutadas = 2 + 8 \times 100 = 802$$

El CPI medio para ejecutar el código anterior será:

$$CPI_{medio} = \frac{Ciclos\ Totales}{Instrucciones\ Ejecutadas} = \frac{1502}{802} = 1.87$$

Nombre: _____

- e) **(0,4 puntos)** Reordena las instrucciones del anterior bucle y rellena el delay slot del salto para que evitar el máximo posible de las paradas y escribe el código resultante.

Solución:

Una posible solución sería llenar el delay slot con la instrucción sub \$4, \$4, \$6 y mover las instrucciones lw \$6, 0(\$2) y addi \$3, \$3, -1 para evitar dependencias de datos:

```
addi $3, $0, 100
addi $4, $0, $0
Loop:
    lw $5, 0($1)#
    lw $6, 0($2)      # Instrucción movida para evitar paradas load
    addi $3, $3, -1   # Instrucción movida para evitar dependencia de datos con bne
    add $4, $4, $5
    addi $1, $1, 4
    addi $2, $2, 4
    bne $3, $0, Loop
    sub $4, $4, $6      # Instrucción que rellena el delay slot
```

Hay que tener en cuenta que serían posibles otras soluciones.

- f) **(0,4 puntos)** Calcula el número de ciclos y CPI medio para ejecutar todas las iteraciones del bucle con el nuevo código. ¿Cuál es la ganancia obtenida?

Solución

Hay 100 iteraciones y cada iteración requiere 8 ciclos ya que no hay paradas.

Hay dos ciclos adicionales por las primeras 2 instrucciones antes del bucle y hay que añadir 4 ciclos adicionales para terminar la instrucción *addi* en la última iteración.

Por tanto:

$$\text{Ciclos Totales} = 100 \times 8 + 6 = 806 \text{ ciclos}$$

El número total de instrucciones ejecutadas no cambia:

$$\text{Instrucciones Ejecutadas} = 2 + 8 \times 100 = 802$$

El CPI medio para ejecutar todas las iteraciones del bucle en este caso será:

$$CPI_{medio} = \frac{\text{Ciclos Totales}}{\text{Instrucciones Ejecutadas}} = \frac{806}{802} = 1,01$$

La ganancia obtenida al reorganizar el código y llenar el delay slot es:

$$\text{Ganancia} = \frac{CPI_{apartado\ d})}{CPI_{apartado\ f})} = \frac{1.87}{1.01} = 1.85$$

Arquitectura de los Computadores. Primera convocatoria 2017

4. (3 puntos) Considera que el segmento de código que aparece a continuación se ejecuta sobre una máquina segmentada MIPS estándar de 5 etapas. Supongamos que el valor inicial del registro R23 es mucho mayor que el valor inicial del registro R20 y que todas las referencias de memoria se encuentran en las cachés y TLBs.

LOOP: lw R10, 0(R20)
 lw R11, 100(R20)
 subu R10, R10, R11
 addiu R20, R20, 4
 sw R10, 200(R20),
 subu R5, R23, R20
 bnez R5, LOOP

- a) (0,5 puntos) Dibuja el diagrama de temporización para la ejecución de la secuencia de instrucciones que se muestran en la tabla. Considera que el salto se resuelve en la etapa ID y que además la técnica del adelantamiento o forwarding no está implementada en esta máquina.

Solución:

Instrucción	Ciclos de reloj																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
lw R10 0(R20)	IF	ID	EX	M	WB																				
lw R11, 0(R20)		IF	ID	EX	M	WB																			
subu R10, R10, R11			IF	ID	P	ID	EX	M	WB																
addiu R20, R20, #4						IF	ID	X	M	WB															
sw R10, 200(R20)							IF	ID	P	ID	EX	M	WB												
subu R5, R23, R20										IF	ID	EX	M	WB											
bnez R5, LOOP											IF	P	P	ID	EX	M	WB								
lw R10, 0(R20)												P	IF	ID	EX	M	WB								

- b) (0,4 puntos) Obtén el número total de ciclos necesarios para completar 2 iteraciones del bucle y calcula el CPI medio obtenido en completar esta ejecución. Muestra cómo has obtenido los valores.

Solución:

Desde la primera búsqueda del primer lw hasta la segunda búsqueda del primer lw han pasado 14 ciclos. Como el bucle itera dos veces serán 28 ciclos. A estos hay que sumar unos pocos ciclos extra para vaciar el pipeline, en concreto 3. En total, el número de ciclos necesarios para completar 2 iteraciones del bucle es de 31 ciclos.

El número de instrucciones ejecutadas en las dos iteraciones son $2 \times 7 = 14$ instrucciones.

Por tanto podemos calcular el CPI medio en completar la ejecución de 2 iteraciones del bucle como:

$$CPI_{medio} = \frac{\text{Número de ciclos}}{\text{Número de instrucciones}} = \frac{31}{14} = 2,21$$

- c) **(0,5 puntos)** Supongamos ahora que se implementa la técnica de *forwarding* en la máquina anterior. Con esta nueva condición rellenar el diagrama de temporización siguiente:

Solución:

Instrucción	Ciclos de reloj																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Lw R10, 0(R20)	IF	ID	EX	M	WB																			
Lw R11, 0(R20)		IF	ID	EX	M	WB																		
subu R10, R10, R11			IF	ID	ID	EX	M	WB																
addiu R20, R20, #4				IF	ID	X	M	WB																
sw R10, 200(R20)					IF	ID	EX	M	WB															
subu R5, R23, R20						IF	ID	EX	M	WB														
bnez R5, LOOP							IF	P	ID	EX	M	WB												
lw R10, 0(R20)								P	IF	ID	EX	M	WB											

- d) **(0,4 puntos)** Obtén el número total de ciclos necesarios para completar 2 iteraciones del bucle. ¿Cuál es ahora el CPI medio obtenido? ¿Cuál es la ganancia obtenida con esta alternativa?

Solución:

Desde la primera búsqueda del primer lw hasta la segunda búsqueda del primer lw han pasado 10 ciclos. Como el bucle itera dos veces serán 20 ciclos. A estos hay que sumar unos pocos ciclos extra para vaciar el pipeline, en concreto 3. En total, el número de ciclos necesarios para completar 2 iteraciones del bucle es de 23 ciclos.

El número de instrucciones ejecutadas en las dos iteraciones no ha cambiado, $2 \times 7 = 14$ instrucciones. Por tanto podemos calcular el CPI medio en completar la ejecución como:

$$CPI_{medio} = \frac{\text{Número de ciclos}}{\text{Número de instrucciones}} = \frac{23}{14} = 1,64$$

La ganancia obtenida con esta implementación se puede calcular como:

$$\text{Ganancia} = \frac{RI_{a)} }{RI_{c)}} = \frac{CPI_{a)}}{CPI_{c)}} = \frac{31}{23} = \frac{2,21}{1,64} = 1,35$$

Es decir, al aplicar la técnica de forwarding el pipeline es un 35% más rápido.

- e) **(0,4 puntos)** Supongamos ahora que el procesador implementa saltos retardados (*delay slot*). ¿Podrías reorganizar el código para evitar cualquier parada, aprovechar el delay slot y conseguir mayor ganancia? Razona la respuesta. Muestra en cualquier caso como llenarías el delay slot.

Solución:

Efectivamente se puede mejorar la ejecución del programa aprovechando el *delay slot* y llenándolo con una instrucción que se ejecute siempre dentro del bucle y no tenga ningún problema de dependencia con la instrucción *bnez*. Además se puede reorganizar el código para evitar la dependencia del registro R5 con la instrucción *bnez*. Los cambios se muestran a continuación:

Numeramos las instrucciones del código original:

LOOP:	lw R10, 0(R20)	;1
	lw R11, 100(R20)	;2
	subu R10, R10, R11	;3
	addiu R20, R20, 4	;4
	sw R10, 200(R20),	;5
	subu R5, R23, R20	;6
	bnez R5, LOOP	;7

Código reorganizado llenando el delay slot:

```
LOOP: lw R10, 0(R20)      ;1  
      lw R11, 100(R20)    ;2  
      addiu R20, R20, 4   ;4  
      subu R10, R10, R11  ;3  
      subu R5, R23, R20   ;6  
      bnez R5, LOOP       ;7  
      sw R10, 200(R20),    ;5
```

Código reorganizado llenando el delay slot y evitando cualquier parada:

```
LOOP: lw R10, 0(R20)      ;1  
      lw R11, 100(R20)    ;2  
      addiu R20, R20, 4   ;4  
      subu R5, R23, R20   ;6  
      subu R10, R10, R11  ;3  
      bnez R5, LOOP       ;7  
      sw R10, 200(R20),    ;5
```

- f) **(0,3 puntos)** ¿Cuál sería en este caso el número de ciclos que tardaría en ejecutarse 2 iteraciones del bucle? ¿Cuál sería la ganancia conseguida para este código respecto a la versión original si la hubiere?

Solución:

Como no hay ninguna parada, desde la primera búsqueda del primer lw hasta la segunda búsqueda del primer lw han pasado 7 ciclos. Como el bucle itera dos veces serán 14 ciclos. A estos hay que sumar unos pocos ciclos extra para vaciar el pipeline, en concreto 3. En total, el número de ciclos necesarios para completar 2 iteraciones del bucle es de 17 ciclos.

El número de instrucciones ejecutadas en las dos iteraciones no ha cambiado, $2 \times 7 = 14$ instrucciones. Por tanto, podemos calcular el CPI medio en completar la ejecución como:

$$CPI_{medio} = \frac{\text{Número de ciclos}}{\text{Número de instrucciones}} = \frac{17}{14} = 1,21$$

La ganancia obtenida reorganizando el código y llenando el delay slot respecto a la versión original será:

$$\text{Ganancia} = \frac{RI_a)}{RI_e)} = \frac{CPI_a)}{CPI_e)} = \frac{31}{17} = \frac{2,21}{1,21} = 1,82$$

Es decir, el código ahora se ejecuta un 82% más rápido que la versión original del apartado a).

- g) **(0,5 puntos)** Se está estudiando dos posibilidades para mejorar la segmentación en los saltos en la máquina anterior en la que el salto se resuelve en la etapa ID. La primera posibilidad es utilizar saltos retardados, en este caso en una ejecución típica del 25% de las instrucciones de salto se consigue llenar el 50% de los delay slot con instrucciones útiles. **Calcula** la ganancia que se consigue al implementar esta técnica. La segunda posibilidad es suponer que los saltos son no efectivos. En este caso en una ejecución típica el 25% de los saltos en un bucle son no efectivos y el 80% de las instrucciones son saltos dentro de un bucle. **Calcula** la ganancia que se consigue al implementar esta técnica. Por simplicidad suponer que las instrucciones que no son de salto su CPI es 1. ¿Qué optimización es la mejor opción?

Solución:

Para calcular la mejora vamos a comparar los CPI del cauce en cada opción que se podrán calcular como:

$$CPI_{segmentación} = CPI_{ideal} + \text{ciclos reloj detención por instrucción}$$

- Mejora introducida con la inclusión de saltos retardados.

Calculemos primero cual es el CPI original, es decir, sin ninguna mejora introducida. En este caso para cada instrucción de salto se debe incluir una parada de un ciclo de reloj, como el 25% de las instrucciones son saltos el CPI se puede calcular como:

$$CPI_{sin\ delay\ slot} = CPI_{ideal} + \text{ciclos reloj detención por instrucción} = 1 + 0,25 \times 1 = 1,25$$

Calculemos ahora cual es el CPI con la mejora introducida al incluir los delay slot. En este caso el CPI será:

$$CPI_{con\ delay\ slot} = CPI_{ideal} + \text{ciclos reloj detención por instrucción} = 1 + 0,5 \times 0,25 \times 1 = 1,125$$

La ganancia conseguida al implementar la técnica de los saltos retardados es:

Nombre: _____

$$Ganancia = \frac{CPI_{\text{sin delay slot}}}{CPI_{\text{con delay slot}}} = \frac{1,25}{1,125} = 1,11$$

- Mejora introducida al suponer que los saltos son no efectivos.

Calculemos primero cual es el CPI original, es decir, sin ninguna mejora introducida. En este caso de nuevo para cada instrucción de salto el CPI se puede calcular como:

$$CPI_{\text{sin mejora}} = CPI_{\text{ideal}} + \text{ciclos reloj detención por instrucción} = 1 + 0,8 \times 1 = 1,8$$

Calculemos ahora el CPI con la mejora introducida al suponer que los saltos son no efectivos:

$$CPI_{\text{saltos no tomados}} = CPI_{\text{ideal}} + \text{ciclos reloj detención por instrucción} = 1 + 0,8 \times 0,75 \times 1 = 1,6$$

La ganancia conseguida al implementar la técnica de suponer saltos no efectivos es:

$$Ganancia = \frac{CPI_{\text{sin mejora}}}{CPI_{\text{saltos no tomados}}} = \frac{1,8}{1,6} = 1,125$$

La mejor opción será la que obtenga mayor ganancia, en este caso se consigue con la introducción de la mejora al suponer que los saltos son no efectivos.

Arquitectura de los Computadores. Primera convocatoria 2017

- 1. (2 puntos)** La empresa CVApps (Computer Vision Applications) está diseñando un sistema de guiado de vehículos mediante metodologías basadas en “deep learning”, cuyos requerimientos de rendimiento vienen impuestos por las altas velocidades de los vehículos. Se ha hecho un estudio de las partes del código del sistema de guiado que son paralelizables y ejecutables en GPUs. El código paralelizable se ha ejecutado en modelos de GPU con diferentes prestaciones:

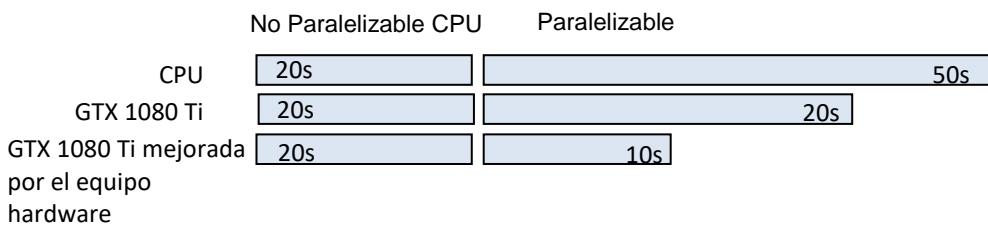
	No Paralelizable CPU	Paralelizable
CPU	20s	50s
GTX 1080 Ti	20s	20s
Radeon Pro Duo	20s	4s
GTX TITAN X	20s	1,67s
GTX 1060, 3 GB	20s	0,33 s

- a) (1 punto)** Calcula la aceleración global y la aceleración mejorada respecto a la opción CPU para cada una de las GPUs

	Tiempo no paralelizable	Tiempo paralelizable	Tiempo total = Tiempo no paralelizable + Tiempo paralelizable	A mejorada respecto a la opción CPU de cada una de las GPUs= Tiempo paralelizable CPU / Tiempo paralelizable GPUs	Aceleración global respecto a la opción CPU de cada una de las GPUs= Tiempo total CPU / Tiempo total GPUs
CPU	20,00	50,00	70,00	50/50	70/70
GTX 1080 Ti	20,00	20,00	40,00	50/20	70/40
Radeon Pro Duo	20,00	4,00	24,00	50/4	70/24
GTX TITAN X	20,00	1,67	21,67	50/1,67	70/21,67
GTX 1060, 3 GB	20,00	0,33	20,33	50/0,33	70/20,33

Ejercicio inspirado en la página 30 del tema 2 de teoría

- b) (1 punto)** El equipo de diseño hardware ha comprobado que es capaz de mejorar la “GTX 1080 Ti” reduciendo a la mitad los tiempos de GPU. Se desea saber si el equipo de desarrollo software puede incrementar el porcentaje de paralelización (fracción mejorada) para mejorar el rendimiento manteniendo la versión actual de la “GTX 1080 Ti”. ¿Qué incremento en el porcentaje de paralelización se necesitará para obtener la misma ganancia de rendimiento que el equipo hardware?



La aceleración global que puede conseguir el equipo hardware es:

$$A_{g\ hardware} = \frac{50 + 20}{20 + 10} = \frac{70}{30} = 2,33$$

El equipo de desarrollo software utiliza la “GTX 1080 Ti” no mejorada, cuya aceleración mejorada es:

$$A_{mejorada} = \frac{50}{20} = 2,5$$

El equipo de desarrollo software se plantea la misma aceleración global que el equipo hardware “2,33”, utilizando la “GTX 1080 Ti” no mejorada cuya aceleración mejorada es “2,5”, incrementando el porcentaje de paralelización o fracción mejorada:

$$A_{g\ hardware} = 2,33 = \frac{1}{(1 - f_{m\ software}) + \frac{f_{m\ software}}{a_m}} = \frac{1}{(1 - f_{m\ software}) + \frac{f_{m\ software}}{2,5}} \rightarrow f_{m\ software} = 0,95$$

La fracción mejorada inicial (siempre definida sobre la opción peor) es:

$$f_{mejorada\ inicial} = \frac{\text{Tiempo paralelizable CPU}}{\text{Tiempo total CPU}} = \frac{50}{50 + 20} = \frac{50}{70} = 0,7143$$

El incremento en el porcentaje de paralelización (fracción mejorada) para obtener la misma ganancia de rendimiento que el equipo hardware (2,33) es:

$$\Delta_{fm} = |f_{m\ software} - f_{m\ inicial}| = |0,95 - 0,7143| = 0,238$$

Ejercicio inspirado en la página 40 del tema 2 de teoría

2. (1,5 puntos) La misma empresa CVApps (Computer Vision Applications) está diseñando un procesador especializado para un sistema de visualización realista, donde es necesario un alto rendimiento en aplicaciones de generación de gráficos. En este procesador y ejecutando estas aplicaciones la mezcla de instrucciones y CPIs son:

Instrucción	Frecuencia	CPI
ALU	40%	1
LOAD	11%	2
STORE	29%	2
SALTO	15%	1
IMP	5%	4

La máquina, debe realizar siempre instrucciones STORE para almacenar los datos que utiliza la instrucción IMP para imprimir en pantalla. La empresa está pensando en realizar una modificación para que IMP cargue directamente los datos a imprimir en pantalla, sin necesidad de realizar antes una STORE. Supongamos que este repertorio extendido de instrucciones incrementa en 1 el número de ciclos de reloj para la instrucción IMP, pero sin afectar a la duración del ciclo de reloj.

- a) Calcula la aceleración de la versión supuestamente mejorada respecto a la anterior

Tiempo de ejecución arquitectura A:

$$T_{ejecución A} = RI_A * CPI_A * clk_A$$

$$CPI_A = 0,4 * 1 + 0,11 * 2 + 0,29 * 2 + 0,15 * 1 + 0,05 * 4 = 1,55$$

$$T_{ejecución A} = RI_A * 1,55 * clk_A$$

Tiempo de ejecución arquitectura mejorada B:

En la arquitectura B, dado que la nueva instrucción IMPn carga directamente los datos a imprimir en pantalla sin necesidad de realizar antes una STORE, es posible eliminar las instrucciones STORE asociadas a instrucciones IMP. Es decir las parejas STORE; IMP de la arquitectura A se remplazan por IMPn. En esta nueva situación la mezcla de instrucciones y CPIs queda de la siguiente forma:

Instrucción	Frecuencia	CPI	Frecuencia modificada	Frecuencia sobre 100%	CPI
ALU	40%	1	40%	42,1%	1
LOAD	11%	2	11%	11,6%	2
STORE	29%	2	29% - 5% = 24%	25,3%	2
SALTO	15%	1	15%	15,8%	1
IMP	5%	4	0%	0%	0
IMPn	0%	0	5%	5,3%	5
	100%		100% - 5% = 95%	100%	

$$T_{ejecución B} = RI_B * CPI_B * clk_B$$

$$CPI_B = \frac{0,4 * 1 + 0,11 * 2 + 0,24 * 2 + 0,15 * 1 + 0,05 * 5}{1 - 0,05} = 0,421 * 1 + 0,116 * 2 + 0,253 * 2 + 0,158 * 1 + 0,053 * 5 = 1,58$$

$$clk_B = clk_A$$

$$RI_B = 0,95 * RI_A$$

$$T_{ejecución B} = RI_B * CPI_B * clk_B = 0,95 * RI_A * 1,58 * clk_A = 1,5 * RI_A * clk_A$$

$$A = \frac{T_{ejecución A}}{T_{ejecución B}} = \frac{1,55 * RI_A * clk_A}{1,5 * RI_A * clk_A} = \frac{1,55}{1,5} = 1,0333$$

Ejercicio inspirado en la página 75 del tema 2 de teoría

Arquitectura de los Computadores. Junio 2012

1. **(3,5 puntos)** La extensión del repertorio Pentium III, incorporando instrucciones SSE y el correspondiente hardware de procesamiento SSE permite acelerar los tiempos de cálculo, en lo que a tareas multimedia en punto flotante (TMPF) se refiere, en un factor de 20. Utilizando como conjunto de benchmarks para análisis del rendimiento multimedia en punto flotante CFP2006 (SPEC CPU2006), cuyos programas realizan tanto tarea multimedia en punto flotante (TMPF) como no multimedia en punto flotante (TNMPF), se observó que los tiempos de ejecución de los programas del conjunto de benchmarks son los que aparecen en la tabla tanto compilados utilizando SSE como sin utilizar SSE.

CFP2006 (Floating Point Component of SPEC CPU2006)	Tiempo ejecución sin SSE Pentium III	Tiempo ejecución con SSE Pentium III
410.bwaves	140 s	60 s
416.gamess	127 s	54 s
433.milc	132 s	58 s
434.zeusmp	114 s	53 s
435.gromacs	154 s	61 s
436.cactusADM	124 s	52 s
437.leslie3d	143 s	59 s
444.namd	154 s	62 s
447.dealII	127 s	55 s
450.soplex	148 s	62 s
453.povray	165 s	68 s
454.calculix	125 s	54 s
459.GemsFDTD	142 s	61 s
465.tonto	144 s	63 s
470.lbm	167 s	70 s
481.wrf	142 s	62 s
482.sphinx3	186 s	72 s
Media geométrica	147,5	60,5

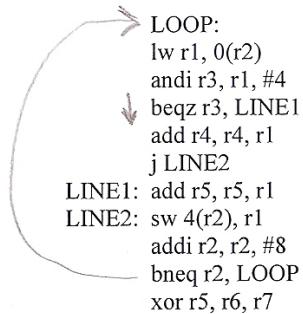
Utilizando la media geométrica para representar el rendimiento de las dos opciones (Tiempo ejecución sin SSE Pentium III; Tiempo ejecución con SSE Pentium III) calcula:

- a) El porcentaje medio del tiempo de ejecución de los programas compilados sin SSE que se utiliza para realizar tareas multimedia en punto flotante (TMPF).
- b) El tiempo de ejecución medio que los programas compilados sin SSE consumen en realizar tareas multimedia en punto flotante (TMPF).
- c) El tiempo de ejecución medio que los programas compilados con SSE consumen en realizar tareas multimedia en punto flotante (TMPF).
- d) El tiempo de ejecución medio que los programas consumen en realizar tareas no multimedia en punto flotante (TNMPF).
- e) Tras estudiar los niveles de utilización de instrucciones SSE en los benchmarks, se estableció que una medida realista de la fracción mejorada es 61%. El departamento de diseño hardware establece la posibilidad de mejorar la unidad SSE duplicando la aceleración mejorada (40). ¿Que incremento en la fracción mejorada sobre el 61% debería conseguir el departamento de diseño de compiladores, para igualar la mejora en la aceleración global conseguida por el departamento de diseño hardware?
2. **(2,5 puntos)** Google está rediseñando su smartphone Nexus con el objetivo de aumentar el rendimiento del teléfono y permita realizar tareas que normalmente estaban en el ámbito de los computadores personales. Para ello, desea rehacer la arquitectura del procesador comenzando por un nuevo diseño del repertorio de instrucciones sabiendo que:
- El coste total del procesador debe ser medio.
 - Los programas que se ejecuten en el procesador se desarrollarán con lenguajes de alto nivel.
 - Tienen pensado una organización del computador con 4 nucleos de procesamiento superescalares.

Indica que decisiones de manera **justificada** debería tomar Intel, **valorando las diferentes alternativas**, en las siguientes componentes de la arquitectura:

- Tipo de almacenamiento interno a la CPU
- Número de operandos que se pueden direccionar en memoria en instrucciones ALU

- c) Modos de direccionamiento de operandos
d) Codificación de los modos de direccionamiento
e) Tipos de instrucciones en el repertorio
f) Tipos de datos, codificación de los mismos y forma de designación del tipo
g) Formas de especificar el destino del salto y la condición de salto
3. (2,5 puntos) Suponed una arquitectura no segmentada que funciona a 2.5 GHz y que necesita 5 ciclos para finalizar la ejecución de una instrucción. Se quiere segmentar la máquina con 5 etapas. Para ello considerad un procesador segmentado con un cauce de 5 etapas. El recuento de instrucciones dinámicas por tipo como un porcentaje del total es el siguiente:
- 10% almacenamientos
20% cargas
20% saltos
50% instrucciones ALU
- a) ¿Cuál es la ganancia ideal debido a la segmentación para este procesador?
- b) Suponed inicialmente que se ignora el incremento potencial del ciclo de reloj debido a la segmentación. En este caso las paradas por dependencia de datos ocurren en dos situaciones. Una parada de un ciclo tiene lugar cuando a una instrucción de carga le sigue una instrucción ALU que utiliza el resultado de la load. Esta situación ocurre en el 40% de las instrucciones de carga. Una parada de 2 ciclos de reloj ocurre cuando a una instrucción de salto le precede una operación ALU cuyo resultado lo utiliza la condición de salto. Esta situación ocurre en el 50% de las instrucciones de salto.
¿Cuál es el porcentaje de reducción de la ganancia respecto a la ideal debido a las dependencias de datos?
- c) Realmente al segmentar la máquina se modifica e incrementa el hardware de manera que se fuerza a correr la máquina a 2 GHz. Suponed que además de las detenciones por dependencias de datos anteriores, se consideran las paradas causadas por las instrucciones de acceso a memoria que provocan una parada de 2 ciclos. Esto ocurre en el 10% de las instrucciones de memoria. ¿Cuál es la ganancia de velocidad en este caso debido a la segmentación?
- d) Muestra el diagrama de temporización en el pipeline de 5 etapas anterior para la ejecución del código siguiente, muestra solo hasta la segunda vez que se haga la búsqueda de la instrucción `Iw`. El primer salto es no efectivo (no se toma) pero el último salto condicional si es efectivo (se toma). El adelantamiento se realiza siempre.



4. (1,5 puntos) La memoria principal del procesador Intel ATOM es de 4GB con un bus de direcciones de 32 bits y palabras de 32 bits. El procesador dispone de una cache de 2MB y dispone de bloques de 64 KB para transferir información entre memoria principal y caché. Indica para cada dirección qué bits corresponden a cada uno de los campos en los que puede dividirse la dirección para:
- a) Correspondencia directa
b) Correspondencia asociativa
c) Correspondencia asociativa por conjuntos de 2 vias

Soluciones

1. (3,5 puntos) La extensión del repertorio Pentium III, incorporando instrucciones SSE y el correspondiente hardware de procesamiento SSE permite acelerar los tiempos de cálculo, en lo que a tareas multimedia en punto flotante (TMPF) se refiere, en un factor de 20. Utilizando como conjunto de benchmarks para análisis del rendimiento multimedia en punto flotante CFP2006 (SPEC CPU2006), cuyos programas realizan tanto tarea multimedia en punto flotante (TMPF) como no multimedia en punto flotante (TNMPF), se observó que los tiempos de ejecución de los programas del conjunto de benchmarks son los que aparecen en la tabla tanto compilados utilizando SSE como sin utilizar SSE.

CFP2006 (Floating Point Component of SPEC CPU2006)	Tiempo ejecución sin SSE Pentium III	Tiempo ejecución con SSE Pentium III
410.bwaves	140 s	60 s
416.ganess	127 s	54 s
433.milc	132 s	58 s
434.zeusmp	114 s	53 s
435.gromacs	154 s	61 s
436.cactusADM	124 s	52 s
437.leslie3d	143 s	59 s
444.namd	154 s	62 s
447.dealll	127 s	55 s
450.soplex	148 s	62 s
453.povray	165 s	68 s
454.calculix	125 s	54 s
459.GemsFDTD	142 s	61 s
465.tonto	144 s	63 s
470.lbm	167 s	70 s
481.wrf	142 s	62 s
482.sphinx3	186 s	72 s
Media geométrica	142 s	60 s

Utilizando la media geométrica para representar el rendimiento de las dos opciones (Tiempo ejecución sin SSE Pentium III; Tiempo ejecución con SSE Pentium III) calcula:

- a) El porcentaje medio del tiempo de ejecución de los programas compilados sin SSE que se utiliza para realizar tareas multimedia en punto flotante (TMPF).
- b) El tiempo de ejecución medio que los programas compilados sin SSE consumen en realizar tareas multimedia en punto flotante (TMPF).
- c) El tiempo de ejecución medio que los programas compilados con SSE consumen en realizar tareas multimedia en punto flotante (TMPF).
- d) El tiempo de ejecución medio que los programas consumen en realizar tareas no multimedia en punto flotante (TNMPF).
- e) Tras estudiar los niveles de utilización de instrucciones SSE en los benchmarks, se estableció que una medida realista de la fracción mejorada es 61%. El departamento de diseño hardware establece la posibilidad de mejorar la unidad SSE duplicando la aceleración mejorada (40). ¿Que incremento en la fracción mejorada sobre el 61% debería conseguir el departamento de diseño de compiladores, para igualar la mejora en la aceleración global conseguida por el departamento de diseño hardware?

Media geométrica:

NO OS FIEIS DE LA SOLUCIÓN

$$\text{sin SSE} \sqrt[n]{\prod_1^n x_i} = \sqrt[17]{140 \cdot 127 \cdot 132 \cdot 114 \dots \cdot 186} = 142,11 \sim 142\%$$

$$\text{sin SSE} \sqrt[n]{\prod_1^n x_i} = \sqrt[17]{60 \cdot 54 \cdot 58 \cdot 53 \cdot \dots \cdot 72} = 60,09 \sim 60\%$$

a)

$$\frac{\text{tiempo}_{\text{sin sse}}}{\text{tiempo}_{\text{con sse}}} = \frac{1}{(1 - f_m) + \frac{f_m}{A_m}} \rightarrow f_m = \text{porcentaje tiempo ejecucion} = \frac{\frac{T_{\text{snsse}} \cdot A_m}{T_{\text{con sse}}} - A_m}{-A_m + 1}$$

Se hace con la media geométrica por tanto será.

$$\frac{142}{60} = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \quad \text{Calculamos} \quad f_m = 0,60785 \sim 60,7\%$$

b)

Teniendo el porcentaje medio en calcular tareas en coma flotante y el tiempo de ejecución sin SSE, solo debemos:

$$tiempo_{\text{sin sse,fot}} = 142 \cdot 60,7\% = 86,194 \text{ s}$$

c)

$$tiempo_{\text{con sse, float}} = \frac{tiempo_{\text{sin sse, float}}}{factor} = \frac{86,194}{20} = 4,31 \text{ s}$$

d) $total - tiempo \text{ para coma flotante} = 142 - 86,194 = 55,81 \text{ s}$

e) Calculamos la aceleración con factor 40 y el 61 %

$$a_g = \frac{1}{(1 - 0,61) + \frac{0,61}{40}} = 2,47$$

con esa ganancia calculamos el fm con el factor 20 antiguo

$$a_g = 2,47 = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \rightarrow f_m = 0,63$$

El incremento por tanto será

$$Inc = 0,63 - 0,61 = 0,02 \sim 2\%$$

2. (2,5 puntos) Google está rediseñando su smartphone Nexus con el objetivo de aumentar el rendimiento del teléfono y permita realizar tareas que normalmente estaban en el ámbito de los computadores personales. Para ello, desea rehacer la arquitectura del procesador comenzando por un nuevo diseño del repertorio de instrucciones sabiendo que:

- i. El coste total del procesador debe ser medio.
- ii. Los programas que se ejecuten en el procesador se desarrollarán con lenguajes de alto nivel.
- iii. Tienen pensado una organización del computador con 4 núcleos de procesamiento superescalares.

Indica que decisiones de manera **justificada** debería tomar Intel, valorando las diferentes alternativas, en las siguientes componentes de la arquitectura:

- a) Tipo de almacenamiento interno a la CPU
- b) Número de operandos que se pueden direccionar en memoria en instrucciones ALU
- c) Modos de direccionamiento de operandos
- d) Codificación de los modos de direccionamiento
- e) Tipos de instrucciones en el repertorio

~~35,2 x 684,7 mm~~ Tipos de datos, codificación de los mismos y forma de designación del tipo

- a) Yo diría almacenamiento R-R, ya que los compiladores pueden aprovechar mucho más estos registros, y son más rápidos que el acceso a memoria.
- b) Registro-registro, sin referencia a memoria. Ya que así aprovecha los registros, más rápidos que la memoria y al encangarse el compilador no supone una carga para el programador.

Las demás no soy seguro(P.D. Las anteriores tampoco XD)

Pregunta 3

$$CLK = 2.5 \text{ GHz} = 2.5 \times 10^9 \text{ Hz}$$

$n = \text{nº ciclos}$ $K = \text{etapas}$

$$\text{Ganancia} = \frac{T_{secrencial}}{T_{segmentado}} = \frac{(n \cdot k) CLK}{(k + (n-1)) CLK}$$

a)

$$G = \frac{(5 \cdot 5) 2.5 \times 10^9}{(5+4) 2.5 \times 10^9} = 2.5 = 272\%$$

b)

* Se ignora el incremento potencial del ciclo de reloj

1 ^a Parada	= 40% de las instrucciones de carga =
1 ciclo	= 8% del total

2 ^a Parada	= 50% instrucciones de salto = 10% del total
2 ciclos	

✗ Porcentaje de reducción de la ganancia ✗

$$1^{\text{a}} \text{ Parada} = \frac{6 \cdot 5}{5+5} = 3 \times 0.08 = 0.24$$

$$2^{\text{a}} \text{ Parada} = \frac{7 \cdot 5}{5+6} = 3.18 \times 0.10 = 0.31$$

$\left. \begin{array}{l} \\ \end{array} \right\} 0.55 = 55\%$

C) Máquina segmentada $CLK = 2 \text{ GHz}$

$$1^{\text{a}} \text{ Parada} = \frac{(6 \cdot 5) 2^5 \cdot 10^6}{(5+5) 2 \cdot 10^6} = 3'75 \times 0'08 = 0'3 = 30\%$$

$$2^{\text{a}} \text{ Parada} = \frac{(7 \cdot 5) 2^5 \cdot 10^6}{(5+6) 2 \cdot 10^6} = 3'92 \times 0'10 = 0'392 = 39'2\%$$

3^a Parada = 10% de las instrucciones de memoria =
= 10% de las instrucciones de carga = 2% del total

$$\frac{(7 \cdot 5) 2^5 \cdot 10^6}{(5+6) 2 \cdot 10^6} = 3'92 \times 0'02 = 0'0794 = 7'94\%$$

$$1^{\text{a}} P + 2^{\text{a}} P + 3^{\text{a}} P = 72,64\%$$

$$G_1 = \frac{(5 \cdot 5) 2^5 \cdot 10^6}{(5+4) 2 \cdot 10^6} = 3'47 = 347\% \\ \underline{- 72'64\%} \\ \boxed{269'36\%}$$

Ejercicio 3 d)

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Loop: lw r1, 0(r2)	IF	ID	EX	MEM	WB																
andi r3, r1, #4		IF	ID	RAW	EX	MEM	WB														
beqz r3, Line1		IF	ID	RAW	RAW	EX	MEM	WB													
halt (parada)		-	-	-	IF	ID	EX	MEM	WB												
add r4,r4,r1						IF	ID	EX	MEM	WB											
j Line 2							IF	ID	EX	MEM	WB										
salto								IF													
Line2: sw 4(r2), r1								IF	ID	EX	MEM	WB									
addi r2, r2, #8									IF	ID	RAW	EX	MEM	WB							
bneq r2, Loop										IF	ID	RAW	RAW	EX	MEM	WB					
salto											IF										
Loop: lw r1, 0(r2)											IF	ID	RAW	RAW	EX	MEM	WB				

Ejercicio 4

4. (1,5 puntos) La memoria principal del procesador Intel ATOM es de 4GB con un bus de direcciones de 32 bits y palabras de 32 bits. El procesador dispone de una cache de 2MB y dispone de bloques de 64 KB para transferir información entre memoria principal y caché. Indica para cada dirección qué bits corresponden a cada uno de los campos en los que puede dividirse la dirección para:

- a) Correspondencia directa
- b) Correspondencia asociativa
- c) Correspondencia asociativa por conjuntos de 2 vias

Formulas

Calculo de la **ETIQUETA** :
$$\frac{\text{Tamaño de memoria principal}}{\text{Tamaño del Bloque}}$$

Calculo de la **LINEA**:
$$\frac{\text{Tamaño Memoria cache}}{\text{Tamaño bloque}}$$

Calculo de la **PALABRA**:
$$\frac{\text{Tamaño del bloque}}{\text{Tamaño palabra}}$$
 ***Pero para estos ejercicios el tamaño de la palabra es = al del bloque.

Calculo de los **CONJUNTOS**:
$$\frac{\text{Número de Lineas}}{\text{Número de vias}}$$

NO OS FIEIS DE LA SOLUCIÓN

****Tanto en correspondencia directa como correspondencia por conjuntos a la etiqueta se le ha de restar la línea(en caso de correspondencia directa) y conjunto(en caso de por conjuntos).

Datos

Memoria principal: 4GB $\rightarrow 2^{32}$

Bus de direcciones: 32bits $\rightarrow 2^5$

Cache: 2MB $\rightarrow 2^{21}$

Bloques: 64KB $\rightarrow 2^{16}$

Palabra = Bloque $\rightarrow 2^{16}$

Para calcular los exponentes utilizamos el $\log_2 N$ donde N es el número que queremos pasar a exponencial de 2

a) Correspondencia directa

$$\text{Etiqueta} = \frac{2^{32}}{2^{16}} - \text{Linea} = 2^{16} - 2^5 = 16 - 5 = 11$$

$$\text{Linea} = \frac{2^{21}}{2^{16}} = 2^5 = 5$$

$$\text{Palabra} = \text{Bloque} = 2^{16} = 16$$

b) Correspondencia asociativa

$$\text{Etiqueta} = \frac{2^{32}}{2^{16}} = 2^{16} = 16$$

$$\text{Palabra} = \text{Bloque} = 2^{16} = 16$$

Los Resultados finales **SON LOS EXPONENTES** de base 2

c) Correspondencia por conjuntos de 2 Vías

$$\text{Etiqueta} = \frac{2^{32}}{2^{16}} - \text{conjuntos} = 2^{16} - 2^4 = 16 - 4 = 12$$

$$\text{Conjuntos} = \frac{32}{2} = 16 = 2^4 = 4$$

$$\text{Palabra} = 2^{16} = 16$$

El 32 sale del número de Líneas que hay $2^5 = 32$ se divide entre las vías.

Resultados finales

- a) 11 - 5 - 16
- b) 16 - 16
- c) 12 - 4 - 16

Arquitectura de los Computadores. Segunda convocatoria 2014

1. (1,75 puntos)

La extensión del repertorio Pentium III, incorporando instrucciones SSE y el correspondiente hardware de procesamiento SSE permite acelerar los tiempos de cálculo, en lo que a tareas multimedia en punto flotante (TMPF) se refiere, en un factor de 20. Utilizando como conjunto de benchmarks para análisis del rendimiento multimedia en punto flotante CFP2006 (SPEC CPU2006), cuyos programas realizan tanto tarea multimedia en punto flotante (TMPF) como no multimedia en punto flotante (TNMPF), se observó que los tiempos de ejecución de los programas del conjunto de benchmarks son los que aparecen en la tabla tanto compilados utilizando SSE como sin utilizar SSE.

CFP2006 (Floating Point Component of SPEC CPU2006)	Tiempo ejecución sin SSE Pentium III	Tiempo ejecución con SSE Pentium III
410 bwaves	140 s	60 s
416.gamess	127 s	54 s
433 milc	132 s	58 s
434 zeusmp	114 s	53 s
435 gromacs	154 s	61 s
436.cactusADM	124 s	52 s
437 leslie3d	143 s	59 s
444 namd	154 s	62 s
447 dealII	127 s	55 s
450 soplex	148 s	62 s
453 povray	165 s	68 s
454.calculix	125 s	54 s
459.GemsFDTD	142 s	61 s
465 tonto	144 s	63 s
470.lbm	167 s	70 s
481 wrf	142 s	62 s
482 sphinx3	186 s	72 s
Media geométrica	142	60

Utilizando la media geométrica para representar el rendimiento de las dos opciones (Tiempo ejecución sin SSE Pentium III. Tiempo ejecución con SSE Pentium III) calcula:

- El porcentaje medio del tiempo de ejecución de los programas compilados sin SSE que se utiliza para realizar tareas multimedia en punto flotante (TMPF). (0,25 puntos)
- El tiempo de ejecución medio que los programas compilados sin SSE consumen en realizar tareas multimedia en punto flotante (TMPF). (0,25 puntos)
- El tiempo de ejecución medio que los programas compilados con SSE consumen en realizar tareas multimedia en punto flotante (TMPF). (0,25 puntos)
- El tiempo de ejecución medio que los programas consumen en realizar tareas no multimedia en punto flotante (TNMPF). (0,25 puntos)
- Tras estudiar los niveles de utilización de instrucciones SSE en los benchmarks, se estableció que una medida realista de la fracción mejorada es 61%. El departamento de diseño hardware establece la posibilidad de mejorar la unidad SSE duplicando la aceleración mejorada (40). ¿Qué incremento en la fracción mejorada sobre el 61% debería conseguir el departamento de diseño de compiladores, para igualar la mejora en la aceleración global conseguida por el departamento de diseño hardware? (0,75 puntos)

2. (1,75 puntos) Supón que estamos considerando dos alternativas para una instrucción de salto condicional,

CPU A. Una instrucción de comparación inicializa un código de condición y es seguida por un salto que examina el código de condición.

CPU B. Se incluye una comparación en el salto.

En ambas CPU, la instrucción de salto condicional emplea 2 ciclos de reloj, y las demás instrucciones 1. En la CPU A, el 20% de todas las instrucciones ejecutadas son saltos condicionales; como cada salto necesita una comparación, otro 20% de las instrucciones son comparaciones. Debido a que la CPU A no incluye la comparación en el salto, su ciclo de reloj es un 25% más rápido que el de la CPU B. ¿Compara los rendimientos de las dos CPUs?

Nombre: _____

3. **(2.5 puntos)** El equipo de diseño de Intel ha pensado rediseñar su arquitectura del repertorio de instrucciones del i7 para contemplar 18 operaciones nuevas a sus 200 operaciones ya existentes. La nueva arquitectura pretende que sus programas sean desarrollados en lenguajes de alto nivel facilitando al compilador su labor. Se plantean tipos de datos enteros, flotantes y cadenas con un acceso a memoria flexible. Los registros serán de 64 bits y, por último, el espacio de direcciones de acceso a memoria es de 16GB.

- a) **(1 punto)** Indica qué decisiones de manera justificada debería tomar el equipo de diseño, valorando las diferentes alternativas, en las siguientes componentes de la arquitectura.

- i. Número de operandos de instrucciones ALU
- ii. Modos de direccionamiento
- iii. Codificación de los modos de direccionamiento
- iv. Tipos de instrucciones en el repertorio
- v. Formas de especificar el destino del salto y la condición de salto

- b) **(1,5 puntos)** Determina los formatos de instrucciones incluyendo los campos de la instrucción y su tamaño según las decisiones tomadas en el apartado anterior

4. **(2.5 puntos)** Suponed que el siguiente fragmento de código se ejecuta en la máquina segmentada de 5 etapas de teoría y que el registro \$a1 tiene inicialmente el valor 2.

LOOP:

```
    lw $t1, 0($a0)
    lw $a0, 0($t1)
    sub $t2, $v0, 3
    sw $t2, 4($a0)
    addi $a1, $a1, -1
    bne $a1, $zero, LOOP
    add $v0, $a0, $zero
    addi $sp, $sp, 8
```

- a) **(0,3 puntos)** Señala todas las posibles dependencias por datos y las instrucciones que causaran riesgos de control durante las dos primeras iteraciones del bucle.
- b) **(0,6 puntos)** Suponed que el fragmento de código se ejecuta en la máquina sin implementación de la técnica de adelantamiento (forwarding) y que el cálculo de la dirección de salto se realiza en la etapa de ejecución. Muestra la ejecución del código en el diagrama de temporización. Utiliza las letras F, D, E, M y W para representar las etapas del cauce, S para las paradas y X para indicar cuando una etapa se "limpia" al descartarsela instrucción. ¿Cuántos ciclos de reloj tardaría el fragmento de código en ejecutarse?
- c) **(0,6 puntos)** Suponed ahora que se adelanta el cálculo de salto a la etapa de decodificación y que el procesador implementa todos los posibles caminos de adelantamiento. Muestra la ejecución del código en el diagrama de temporización suponiendo que se predicen los saltos como no efectivos. ¿Cuántos ciclos de reloj tardaría el fragmento de código en ejecutarse?
- d) **(0,5 puntos)** Suponed ahora que además de forwarding se utiliza la técnica de saltos retardados (delay slot). Si es posible, **reordena** la secuencia de instrucciones para evitar el máximo número de paradas y que se ejecute el fragmento de código en el mínimo número de ciclos de reloj posible. ¿En cuántos ciclos de reloj se ejecutaría el nuevo código?
- e) **(0,5 puntos)** Suponed que previamente a segmentar la máquina anterior, la etapa más larga necesitaba 0.8ns y que el retardo de los registros de segmentación requeridos para segmentarla era de 0.2ns. Determina el CPI del bucle para los apartados b, c y d anteriores y calcula el tiempo medio de ejecución por instrucción para cada caso. ¿En qué porcentaje aumenta la ganancia en cada caso respecto a la máquina del apartado b?
5. **(1,5 puntos)** Supongamos que la penalización de fallos de la caché es de 250 ciclos de reloj y que todas las instrucciones normalmente emplean 2 ciclos de reloj (ignorando las detenciones de memoria). Suponer que el número medio de fallos a la caché por cada 100 instrucciones es 2.
- a) **(1 punto)** ¿Cuál es el impacto en el rendimiento del sistema?
- b) **(0,5 puntos)** Calcula la ganancia de rendimiento que se obtiene, si se amplia el tamaño del bloque de la caché y logramos que el número medio de fallos a la caché sea de 2 cada 1000 instrucciones.

Soluciones

1. (3,5 puntos) La extensión del repertorio Pentium III, incorporando instrucciones SSE y el correspondiente hardware de procesamiento SSE permite acelerar los tiempos de cálculo, en lo que a tareas multimedia en punto flotante (TMFP) se refiere, en un factor de 20. Utilizando como conjunto de benchmarks para análisis del rendimiento multimedia en punto flotante CFP2006 (SPEC CPU2006), cuyos programas realizan tanto tarea multimedia en punto flotante (TMFP) como no multimedia en punto flotante (TNMPF), se observó que los tiempos de ejecución de los programas del conjunto de benchmarks son los que aparecen en la tabla tanto compilados utilizando SSE como sin utilizar SSE.

CFP2006 (Floating Point Component of SPEC CPU2006)	Tiempo ejecución sin SSE Pentium III	Tiempo ejecución con SSE Pentium III
410.bwaves	140 s	60 s
416.gamess	127 s	54 s
433.milc	132 s	58 s
434.zeusmp	114 s	53 s
435.gromacs	154 s	61 s
436.cactusADM	124 s	52 s
437.leslie3d	143 s	59 s
444.namd	154 s	62 s
447.deallII	127 s	55 s
450.soplex	148 s	62 s
453.povray	165 s	68 s
454.calculix	125 s	54 s
459.GemsFDTD	142 s	61 s
465.tonto	144 s	63 s
470.lbm	167 s	70 s
481.wrf	142 s	62 s
482.sphinx3	186 s	72 s
Media geométrica	142 s	60 s

Utilizando la media geométrica para representar el rendimiento de las dos opciones (Tiempo ejecución sin SSE Pentium III; Tiempo ejecución con SSE Pentium III) calcula:

- a) El porcentaje medio del tiempo de ejecución de los programas compilados sin SSE que se utiliza para realizar tareas multimedia en punto flotante (TMFP).
- b) El tiempo de ejecución medio que los programas compilados sin SSE consumen en realizar tareas multimedia en punto flotante (TMFP).
- c) El tiempo de ejecución medio que los programas compilados con SSE consumen en realizar tareas multimedia en punto flotante (TMFP).
- d) El tiempo de ejecución medio que los programas consumen en realizar tareas no multimedia en punto flotante (TNMPF).
- e) Tras estudiar los niveles de utilización de instrucciones SSE en los benchmarks, se estableció que una medida realista de la fracción mejorada es 61%. El departamento de diseño hardware establece la posibilidad de mejorar la unidad SSE duplicando la aceleración mejorada (40). ¿Qué incremento en la fracción mejorada sobre el 61% debería conseguir el departamento de diseño de compiladores, para igualar la mejora en la aceleración global conseguida por el departamento de diseño hardware?

Media geométrica:

NO OS FIEIS DE LA SOLUCIÓN

$$\text{sin SSE} \sqrt[n]{\prod_1^n x_i} = \sqrt[17]{140 \cdot 127 \cdot 132 \cdot 114 \cdots \cdot 186} = 142,11 \sim 142\%$$

$$\text{sin SSE} \sqrt[n]{\prod_1^n x_i} = \sqrt[17]{60 \cdot 54 \cdot 58 \cdot 53 \cdots \cdot 72} = 60,09 \sim 60\%$$

a)

$$\frac{\text{tiempo}_{\text{sin SSE}}}{\text{tiempo}_{\text{con SSE}}} = \frac{1}{(1 - f_m) + \frac{f_m}{A_m}} \rightarrow f_m = \text{porcentaje tiempo ejecución} = \frac{\frac{T_{\text{sin SSE}} \cdot A_m}{T_{\text{con SSE}}} - A_m}{-A_m + 1}$$

Se hace con la media geométrica por tanto será.

$$\frac{142}{60} = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \quad \text{Calculamos } f_m = 0,60785 \sim 60,7\%$$

b)

Teniendo el porcentaje medio en calcular tareas en coma flotante y el tiempo de ejecucion sin SSE, solo debemos:

$$tiempo_{\text{sin.sse.fot}} = 142 \cdot 60,7\% = 86,194 \text{ s}$$

c)

$$tiempo_{\text{consse,float}} = \frac{tiempo_{\text{sin.sse.float}}}{factor} = \frac{86,194}{20} = 4,31 \text{ s}$$

d) $total - tiempo \text{ para coma flotante} = 142 - 86,194 = 55,81 \text{ s}$

e) Calculamos la aceleración con factor 40 y el 61 %

$$a_g = \frac{1}{(1-0,61)+\frac{0,61}{40}} = 2,47$$

con esa ganancia calculamos el fm con el factor 20 antiguo

$$a_g = 2,47 = \frac{1}{(1-f_m)+\frac{f_m}{20}} \rightarrow f_m = 0,63$$

El incremento por tanto será

$$Inc = 0,63 - 0,61 = 0,02 \sim 2\%$$

2. (2,5 puntos) Google está rediseñando su smartphone Nexus con el objetivo de aumentar el rendimiento del teléfono y permita realizar tareas que normalmente estaban en el ámbito de los computadores personales. Para ello, desea rehacer la arquitectura del procesador comenzando por un nuevo diseño del repertorio de instrucciones sabiendo que:

- i. El coste total del procesador debe ser medio.
- ii. Los programas que se ejecuten en el procesador se desarrollarán con lenguajes de alto nivel.
- iii. Tienen pensado una organización del computador con 4 nucleos de procesamiento superescalares.

Indica que decisiones de manera **justificada** debería tomar Intel, valorando las **diferentes alternativas**, en las siguientes componentes de la arquitectura:

- a) Tipo de almacenamiento interno a la CPU
- b) Número de operandos que se pueden direccionar en memoria en instrucciones ALU
- c) Modos de direccionamiento de operandos
- d) Codificación de los modos de direccionamiento
- e) Tipos de instrucciones en el repertorio

35,2 x 68,47 mm Tipos de datos, codificación de los mismos y forma de designación del tipo

- a) Yo diría almacenamiento R-R, ya que los compiladores pueden aprovechar mucho más estos registros, y son mas rápidos que el acceso a memoria.
- b) Registro-registro, sin referencia a memoria. Ya que así aprovecha los registros, más rápidos que de memoria y al encangarse el compilador no supone una carga para el programador.

Las demás no soy seguro(P.D. Las anteriores tampoco XD)

Ejemplo

Suponer que estamos considerando dos alternativas para una instrucción de salto condicional:

CPU A. Una instrucción de comparación inicializa un código de condición y es seguida por un salto que examina el código de condición.

CPU B. Se incluye una comparación en el salto.

En ambas CPU, la instrucción de salto condicional emplea 2 ciclos de reloj, y las demás instrucciones 1. (Obviamente, si el CPI es 1,0 excepto en los saltos de este sencillo ejemplo, estamos ignorando las pérdidas debidas al sistema de memoria; ver la falacia de la pág. 77.) En la CPU A, el 20 por 100 de todas las instrucciones ejecutadas son saltos condicionales; como cada salto necesita una comparación, otro 20 por 100 de las instrucciones son comparaciones. Debido a que la CPU A no incluye la comparación en el salto, su ciclo de reloj es un 25 por 100 más rápido que el de la CPU B. ¿Qué CPU es más rápida?

Respuesta

Como ignoramos todas las prestaciones del sistema, podemos utilizar la fórmula del rendimiento de la CPU: CPI_A es $((0,20 \cdot 2) + (0,80 \cdot 1))$ o 1,2, ya que el 20 por 100 son saltos que emplean 2 ciclos de reloj y el resto 1. La duración de ciclo de reloj_B es $1,25 \cdot \text{Duración ciclo de reloj}_A$, ya que A es un 25 por 100 más rápido. El rendimiento de la CPU A es entonces

$$\begin{aligned}\text{Tiempo CPU}_A &= \text{Recuento de instrucciones}_A \cdot 1,2 \cdot \text{Duración ciclo de reloj}_A \\ &= 1,20 \cdot \text{Recuento de instrucciones}_A \cdot \text{Duración ciclo de reloj}_A\end{aligned}$$

Las comparaciones no se ejecutan en la CPU B, por tanto 20/80 o el 25 por 100 de las instrucciones ahora son bifurcaciones, que emplean 2 ciclos de reloj, y el 75 por 100 restante emplean 1 ciclo. CPI_B es entonces $((0,25 \cdot 2) + (0,75 \cdot 1))$ o 1,25. Como la CPU B no ejecuta comparaciones, la Cuenta de instrucciones_B es $0,80 \cdot \text{Cuenta de instrucciones}_A$. El rendimiento de la CPU B es

$$\begin{aligned}\text{Tiempo CPU}_B &= (0,80 \cdot \text{Recuento de instrucciones}_A) \cdot 1,25 \cdot (1,25 \cdot \text{Duración ciclo de reloj}_A) \\ &= 1,25 \cdot \text{Recuento de instrucciones}_A \cdot \text{Duración ciclo de reloj}_A\end{aligned}$$

Bajo estas hipótesis, la CPU A, con un ciclo de reloj más corto, es más rápida que la CPU B, que ejecuta un número menor de instrucciones.

Julio 2014

⑤

$$PF = 250 \text{ dK}$$

Cuota de Instrucciones = 2 todos los Instrucciones (100%)

$$\frac{\text{fallo}}{\text{nº Instrucciones}} = \frac{2}{100}$$

Formula

$$TCPU = NI \cdot (CPI + \frac{\text{fallo}}{\text{nº Instr}} \cdot PF) \cdot Trel$$

a)

$$TCPU = NI \cdot (2 \cdot 1 + \frac{2}{100} \cdot 250) \cdot Trel = NI \cdot 7 \cdot Trel$$

Hay un Incremento de $\frac{1}{7}$

$$b) TCPU = NI \cdot (2 \cdot 1 + \frac{2}{1000} \cdot 250) \cdot Trel = NI \cdot 2,5 \cdot Trel$$

$$\frac{1}{2,5} = 1 + \frac{n}{100}$$

$$n = 180\%$$

Nombre: _____

Arquitectura de los Computadores. Primera convocatoria 2014

1. (1,75 puntos)

Supongamos que estamos considerando la transformación de una arquitectura carga/almacenamiento (R-R) en una máquina R-M, con objeto de mejorar el rendimiento reduciendo el recuento de instrucciones. Para ello se está estudiando la incorporación de instrucciones aritméticas con una referencia a memoria ALU(R-M) además de las que operan con registros ALU(R-R). La idea es sustituir las secuencias de instrucciones LOAD; ALU(R-R); por ALU(R-M)* para reducir el número de instrucciones LOAD. El compilador realizará estas sustituciones en caso de que el operando cargado y utilizado por una instrucción ALU(R-R) no se utilice de nuevo, situación que se produce con una frecuencia del 30% de las instrucciones ALU(R-R) de la CPU carga almacenamiento. Estas nuevas instrucciones de registro memoria emplean dos ciclos de reloj. En la tabla se observan los CPI por tipo de instrucción de las dos CPUs.

- a) ¿Calcula la aceleración en caso de no haber contrapartida, ni en CPI por tipo de instrucción, ni en clk?
- b) ¿Calcula la aceleración en caso de encontrar la contrapartida del incremento del clk de la CPU B en un 10%?

* LOAD R1,0(R7)
ALU(R-R) R2, R1
Por

ALU(R-M) R2, 0(R7)

CPU A (R-R)		
Operación	Frecuencia	Cuenta de ciclos de reloj
ALU (R-R)	45%	1
Cargas	20%	2
Almacenamientos	10%	2
Saltos	25%	2

CPU B (R-M)		
Operación	Frecuencia	Cuenta de ciclos de reloj
ALU (R-R)	i?	1
ALU (R-M)	i?	2
Cargas	i?	2
Almacenamientos	i?	2
Saltos	i?	2

2. (1,75 puntos) Para las cuatro siguientes preguntas, supongamos que se está considerando mejorar una máquina añadiéndole una unidad aritmética segmentada (UAS). Cuando se ejecuta un cálculo mediante la UAS, es 20 veces más rápido que en el modo normal de ejecución. Llamamos al porcentaje de tiempo que puede emplearse el modo UAS porcentaje UAS.

- a) Dibujar un gráfico donde se muestre la aceleración en relación al porcentaje UAS. Rotular el eje y con "aceleración neta" y el eje x con "porcentaje UAS".
- b) ¿Qué porcentaje UAS se necesita para conseguir una aceleración de 2?
- c) ¿Qué porcentaje UAS se necesita para conseguir la mitad de la aceleración máxima alcanzable utilizando el modo UAS?
- d) Supongamos que hemos medido el porcentaje UAS de programas, obteniendo que es del 70%. El grupo de diseño hardware dice que puede duplicar la velocidad de la UAS con una inversión significativa de ingeniería adicional. Se desea saber si el equipo de compilación puede incrementar la utilización del modo UAS como otra aproximación para incrementar el rendimiento. ¿Qué incremento en el porcentaje UAS (relativo a la utilización actual) se necesitará para obtener la misma ganancia de rendimiento?

3. (2,5 puntos) El mismo equipo de diseño de la pregunta 1, ha pensado rediseñar su arquitectura del repertorio de instrucciones para contemplar la máquina R-M con una referencia a memoria (CPU B). A partir del análisis realizado, se contempla que sean 48 el número de operaciones a realizar incluyendo las aritméticas, lógicas, saltos y referencias a memoria que permiten abordar las tareas de un procesador de propósito general. Dado que los programas que se ejecuten en el procesador se desarrollarán con lenguajes de alto nivel, se contempla un banco de 32 registros de 32 bits. Por último, el espacio de direcciones de acceso a memoria es de 4GB.

- a) **(1 puntos)** Indica qué decisiones de manera **justificada** debería tomar el equipo de diseño, valorando las diferentes alternativas, en las siguientes componentes de la arquitectura.
 - i. Número de operandos de instrucciones ALU
 - ii. Modos de direccionamiento
 - iii. Codificación de los modos de direccionamiento
 - iv. Tipos de instrucciones en el repertorio
 - v. Formas de especificar el destino del salto y la condición de salto

- b) **(1,5 puntos)** Determina los formatos de instrucciones incluyendo los campos de la instrucción y su tamaño según las decisiones tomadas en el apartado anterior

4. (1,5 puntos) Suponer que el siguiente fragmento de código se ejecuta en la máquina segmentada de 5 etapas de teoría con un valor inicial para R8 de R2+16:

Loop: LW R5, 0(R2)
LW R4, 0(R3)
ADD R5, R5, R4
ADD R6, R2, R6
SW R5, 0(R6)
ADDI R2, R2, 4
ADDI R3, R3, 4
SUB R7, R8, R2
BNZ R7, LOOP

Nombre: _____

- a) (0.3 puntos) Suponed que el fragmento de código se ejecuta en la máquina sin implementación de la técnica de adelantamiento (forwarding) y que el cálculo de la dirección de salto se realiza en la etapa de ejecución. ¿Cuántos ciclos de reloj tardaría el fragmento de código en ejecutarse?

b) (0.4 puntos) Suponed ahora que se adelanta el cálculo de salto a la etapa de decodificación y que el procesador implementa todos los posibles caminos de adelantamiento. Muestra la ejecución del código en el diagrama de temporización siguiente suponiendo que se predicen los saltos como no efectivos. ¿Cuántos ciclos de reloj tardaría el fragmento de código en ejecutarse? (Añade más filas en el diagrama si los necesitas).

- c) (0.4 puntos) Suponed ahora que además de forwarding se utiliza la técnica de saltos retardados (delay slot). Si es posible, **reordena** la secuencia de instrucciones para evitar el máximo número de paradas y que se ejecute el fragmento de código en el mínimo número de ciclos de reloj posible. ¿En cuántos ciclos de reloj se ejecutaría el nuevo código?

d) (0,4 puntos) Suponed que la etapa de segmentación más larga necesita 0.8ns y que el retardo de los registros de segmentación es de 1ns. Determina el CPI del bucle para los apartados b y c anteriores y calcula el tiempo medio de ejecución por instrucción para cada caso.

5. (1 punto) Considerad un versión simplificada del MIPS en la que los datos inmediatos de las instrucciones LW y SW han de ser cero. Es decir la instrucción `lw $t0, 0($t1)` es correcta pero no así la instrucción `lw $t0, 4($t1)`. Esta nueva máquina se puede implementar con cuatro etapas de segmentación: IF, ID, EM, WB donde la etapa EM realiza en paralelo las tareas que se hacen en las etapas EX y MEM.

a) (0.4 puntos) Considerad la secuencia de instrucciones :

```
lw $t0, 0($a0)  
lw $t0, 0($t0)
```

Suponiendo que hay adelantamiento, rellena el diagrama de temporización siguiente dónde se señalen (si hay) las paradas para resolver los riesgos de datos y los adelantamientos.

- b) (0.3 puntos) Si la instrucción sw se sustituye por una instrucción de salto condicional, ¿Bajo qué circunstancias sería necesario realizar una parada entre las dos instrucciones?

c) (0.3 puntos) Si la instrucción sw se reemplazara por addi \$t0, \$0, 1 y no estuviera implementado la técnica de adelantamiento. ¿Cuántas paradas serían necesarias? Razona la respuesta

6. **(1,5 puntos)** El sistema de memoria del ejercicio 1 cuenta con un sistema ideal de acceso a la misma. Supongamos en este ejercicio una jerarquía de memoria real con una caché de 2MB que tiene una penalización de fallos de la caché de 150 ciclos de reloj. A partir de los análisis del equipo de diseño, la frecuencia de fallos para ambos sistemas es de 5%. La CPU A tiene una media de 1.25 referencias a memoria por instrucción mientras que la CPU B una media de 1.5 referencias por instrucción (al incluir las instrucciones R-M) ¿Cuál es el impacto en el rendimiento de ambos sistemas cuando se incluye el comportamiento de la caché? Calcularlo para el caso en el que ambas máquinas tienen la misma duración del ciclo de reloj (clk)

CPU A (R-R)

Operación	Frecuencia	Cuenta de Reloj
ALU (R+R)	45%	1
Carga	20%	2
Almacenamiento	10%	2
Salida	25%	2

CPU B (R-M)

ALU (R-R)	$45 * 0.7 = 31.5$	1
ALU (R-M)	$45 * 0.3 = 13.5$	2
Carga	$20 - 0.3 + 0.45 = 6.5$	2
Almacenamiento	10%	2
Salida	25%	2

$$CPI_A = 0.45 + 0.2 \cdot 2 + 0.11 \cdot 2 + 0.25 \cdot 2 = 1.55$$

$$CPI_B = \frac{0.315 + 0.135 \cdot 2 + 0.065 \cdot 2 + 0.1 \cdot 2 + 0.25 \cdot 2}{0.135} = 1.635$$

$$T_{CPU_A} = RI_A \cdot CPI_A \cdot T_{clk}$$

$$\frac{T_{CPU_B}}{T_{CPU_A}} = \frac{0.86 \cdot 1.635}{1.55} =$$

$$T_{CPU_B} = 0.86 RI_A \cdot 1.635 T_{clk}$$

Dar la vuelta.

- ◆ **Ejercicio 3:** Para las cuatro siguientes preguntas, supongamos que se está considerando mejorar una máquina añadiéndole un modo vectorial. Cuando se ejecuta un cálculo en modo vectorial, es 20 veces más rápido que en el modo normal de ejecución. Llamamos al porcentaje de tiempo que puede emplearse el modo vectorial porcentaje de vectorización.
 - ◆ 1. Dibujar un gráfico donde se muestre la aceleración como porcentaje del cálculo realizado en modo vectorial. Rotular el eje y con "aceleración neta" y el eje x con "porcentaje de vectorización".
 - ◆ 2. ¿Que porcentaje de vectorización se necesita para conseguir una aceleración de 2?
 - ◆ 3. ¿Que porcentaje de vectorización se necesita para conseguir la mitad de la aceleración máxima alcanzable utilizando el modo vectorial?.
 - ◆ 4. Supongamos que hemos medido el porcentaje de vectorización de programas, obteniendo que es del 70%. El grupo de diseño hardware dice que puede duplicar la velocidad de la parte vectorizada con una inversión significativa de ingeniería adicional. Se desea saber si el equipo de compilación puede incrementar la utilización del modo vectorial como otra aproximación para incrementar el rendimiento. ¿Que incremento en el porcentaje de vectorización (relativo a la utilización actual) se necesitará para obtener la misma ganancia de rendimiento? ¿Que inversión es recomendable?.

- 1. Dibujar un gráfico donde se muestre la aceleración como porcentaje del cálculo realizado en modo vectorial. Rotular el eje y con "aceleración neta" y el eje x con "porcentaje de vectorización".

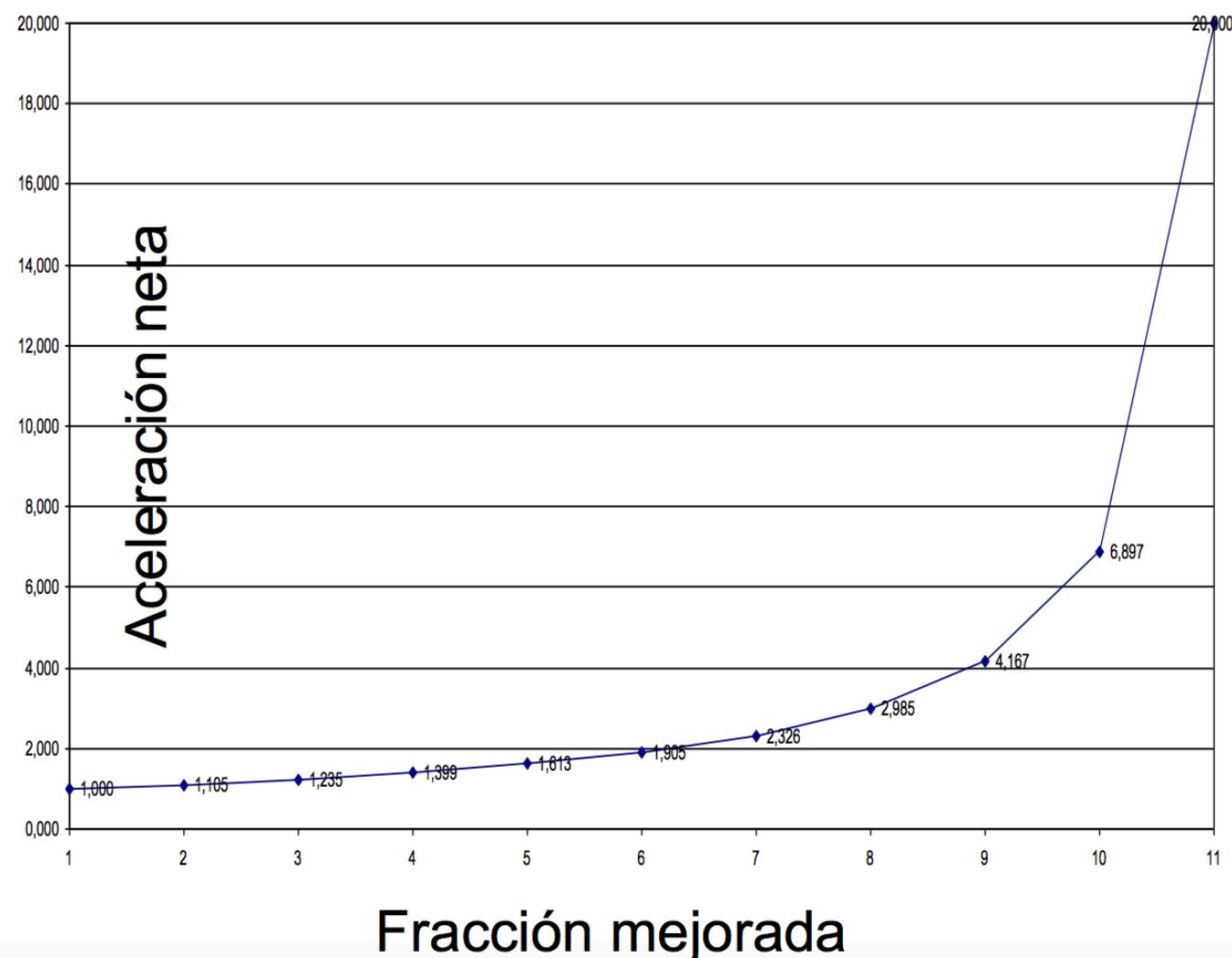
0% $a_g = \frac{1}{1 + \frac{0}{20}} = 1$

10% $a_g = \frac{1}{0,9 + \frac{0,1}{20}} = 1,105$

20% $a_g = \frac{1}{0,8 + \frac{0,2}{20}} = 1,235$

90% $a_g = \frac{1}{0,1 + \frac{0,9}{20}} = 6,9$

100% $a_g = \frac{1}{0 + \frac{1}{20}} = 20$



- ➊ **Ejercicio 3:**

- ➋ 2. ¿Qué porcentaje de vectorización se necesita para conseguir una aceleración de 2?

$$2 = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \Rightarrow f_m = 0,526 = 52,6\%$$

- ➌ 3. ¿Qué porcentaje de vectorización se necesita para conseguir la mitad de la aceleración máxima alcanzable utilizando el modo vectorial?.

$$10 = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \Rightarrow f_m = 0,947 \blacksquare 95\%$$

- 4. Hemos medido el porcentaje de vectorización de programas, obteniendo que es del 70%. El grupo de diseño hardware dice que puede duplicar la velocidad de la parte vectorizada con una inversión significativa de ingeniería adicional. Se desea saber si el equipo de compilación puede incrementar la utilización del modo vectorial como otra aproximación para incrementar el rendimiento. ¿Que incremento en el porcentaje de vectorización (relativo a la utilización actual) se necesitará para obtener la misma ganancia de rendimiento? ¿Que inversión es recomendable?.

$$TE_{ant} = TE_{vec} * 20 = TE_{vec.r} * 40 \rightarrow TE_{vec} = TE_{vec.r} * 2$$

- Aceleración global conseguida por el grupo de hardware

$$a_g = \frac{1}{(1-0,7) + \frac{0,7}{40}} = 3,15$$

- 4. Hemos medido el porcentaje de vectorización de programas, obteniendo que es del 70%. El grupo de diseño hardware dice que puede duplicar la velocidad de la parte vectorizada con una inversión significativa de ingeniería adicional. Se desea saber si el equipo de compilación puede incrementar la utilización del modo vectorial como otra aproximación para incrementar el rendimiento. ¿Que incremento en el porcentaje de vectorización (relativo a la utilización actual) se necesitará para obtener la misma ganancia de rendimiento? ¿Que inversión es recomendable?.

$$TE_{ant} = TE_{vec} * 20 = TE_{vec.r} * 40 \rightarrow TE_{vec} = TE_{vec.r} * 2$$

- Incremento del % de vectorización para alcanzar $a_g=3,15$

$$a_g = 3,15 = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \Rightarrow f_m = 0,719$$

$$\Delta \% vector = |0,7 - 0,719| = 0,019 = 1,9\%$$

Junio 2014

⑥

$$\text{Cache} = 2 \text{ NB}$$

$$\text{PF} = 150 \text{ ns}$$

$$\text{FF} = 5\%$$

$$\frac{\text{NM}}{\text{NI}}_A = 1'25$$

$$\frac{\text{NM}}{\text{NI}}_B = 1'5$$

Del ejercicio +

$$\text{CPI}_A = 1'55$$

$$\text{CPI}_B = 1'67$$

Formule

$$T_{\text{CPU}} = \text{NI} * \left(\text{CPI} + \frac{\text{NM}}{\text{NI}} * \text{FF} * \text{PF} \right) * \text{Trel}$$

$$T_{\text{CPU}_A} = \text{NI} * (1'55 + 1'25 * 0'05 * 150) * \text{Trel} = \\ = \text{NI} * 10'9$$

$$T_{\text{CPU}_B} = \text{NI} * (1'67 + 1'5 * 0'05 * 150) * \text{Trel} = \\ = \text{NI} * 12,9$$

El Rendimiento de CPU_B es peor porque

$$T_{\text{CPU}_B} > T_{\text{CPU}_A}$$

Ejercicio 1 junio 2015

La extensión del repertorio de un microprocesador incorporando instrucciones SSE y el correspondiente hardware de procesamiento SSE permite acelerar los tiempos de cálculo, en lo que a tareas en punto flotante(TMPF) se refiere, en un factor de 6. Utilizando como benchmark para análisis del rendimiento en punto flotante el programa alvinn de SPEC, que realiza tanto tarea TMPF como no multimedia (TNMPF), se observó que el tiempo de ejecución del programa era de 15 segundos si se compilaba utilizando SSE y de 24 segundos si se compilaba sin utilizar SSE

- a. Calcula la aceleración global y la fracción mejorada.

$$A_{global} = \frac{TE_{antiguo}}{TE_{nuevo}} = \frac{24}{16} = 1.6 = \frac{1}{1 - F_m + \frac{F_m}{6}}$$

despejando F_m sale:

$$F_m = 0.45 = 45\%$$

$$A_{global} = 1.6$$

- b. Calcula el tiempo de ejecución que el programa compilado sin SSE consume en realizar tareas multimedia en punto flotante.

$$24 \cdot 0.45 = 10.8s$$

- c. Calcula el tiempo de ejecución que el programa compilado con SSE consume en realizar tareas multimedia en punto flotante.

$$16 - 13.2 = 2.8s$$

- d. Calcula el tiempo de ejecución que el programa consume en realizar tareas no multimedia en punto flotante.

En este caso el tiempo será el mismo tanto para compilación utilizando SSE como sin utilizar SSE.

Se puede calcular de las dos formas

$$24 \cdot 0.55 = 13.2$$

$$24 - 10.8 = 13.2$$

Ejercicio 2 Examen AC junio 2015

2.- En el diseño de una CPU se está considerando la conveniencia de remplazar las instrucciones de salto condicional por una versión que incluya la comparación en el salto. En la alternativa de diseño de la CPU A se han utilizado códigos de condición que son actualizados por la instrucción de comparación CMP. En la alternativa de la CPU B la arquitectura es la misma salvando que las instrucciones de salto realizan la comparación SLTCMP, eliminando las correspondientes instrucciones de comparación y reduciendo el RI. La incorporación de la instrucción SLTCMP incrementa el clk en la CPU B en un 5%. El CPI y frecuencia por tipo de instrucción se detalla en la tabla para la CPU A.

Tipo de instrucción	CPI CPU A	%	CPI CPU B	%
Load/Store	2	35%	2	?
ALU	1	30%	1	?
CMP	1	20%	1	?
SLT (Salto)	2	15%	No existe	
SLTCMP (Compara y salta)	No existe		2	?

- a) Calcula la aceleración entre las dos opciones de diseño.

La aceleración se calcula como el tiempo de ejecución entre una CPU y otra:

$$\frac{T_{ejA}}{T_{ejB}} = \frac{RI_A * CPI_A * CLK_A}{RI_B * CPI_B * CLK_B}$$

El recuento de instrucciones en la CPU B será el mismo que el de la A pero quitándole las instrucciones CMP que acompañaban a las instrucciones SLT (Ahora serán SLTCMP): $RI_B = (1_{TOTAL} - 0'15_{CMP}) * RI_A = 0'85 * RI_A$

El CLK de B es un 5% mayor que el de A: $CLK_B = CLK_A + 0'05CLK_A = 1'05CLK_A$

El CPI medio de A lo calculamos como el CPI individual de cada instrucción por el % que es utilizada: $CPI_A = 2 \cdot 0'35 + 1 \cdot 0'30 + 1 \cdot 0'20 + 2 \cdot 0'15 = 1'50$

Para calcular el CPI de B, debemos de conocer el porcentaje de uso de cada instrucción. Como sabemos que el CPU B no ejecuta las instrucciones de comparación porque las incluye en el salto (pero solo deja de ejecutar aquellas que iban acompañadas de la instrucción SLT), y que la cantidad de saltos con comparación que realizará será la misma que los saltos en la CPU A, deducimos que el 15% de los saltos, el 30% de ALU, el 35% de Load/Store y el 5% de CMP sin salto restantes suponen un 85% (100% total – 15% saltos) de las instrucciones del CPU B. Así, deducimos lo siguiente:

$$L\&S_{CPUB} = \frac{35}{85} = 0'412 ALU_{CPUB} = \frac{30}{85} = 0,353 CMP_{CPUB} = \frac{5}{85} = 0,059 SLTCMP_{CPUB} = \frac{15}{85} = 0,176$$

Ahora ya podemos calcular el CPI medio de B: $CPI_B = 2 \cdot 0'412 + 1 \cdot 0'353 + 1 \cdot 0'059 + 2 \cdot 0'176 = 1'588$

Con esto, podemos sustituir los valores en la expresión inicial:

$$\frac{RI_A * CPI_A * CLK_A}{RI_B * CPI_B * CLK_B} = \frac{RI_A * 1'5 * CLK_A}{0'85 RI_A * 1'588 * 1'05 CLK_A} = \frac{1'5}{0'85 * 1'588 * 1'05} = 1'058$$

Por tanto, la ganancia es de 1'058 o, lo que es lo mismo, un 5'8%.

- b) Calcula el % de incremento del ciclo de reloj a partir del cual el cambio deja de ser rentable.

El cambio deja de ser rentable cuando la ganancia es nula:

$$\frac{1'5}{0'85 * 1'588 * x} = 1 \rightarrow x = \frac{1'5}{0'85 * 1'588} = 1'1112 = 11,12\%$$

Propuesta ejercicio 3 junio 2014 AC

El mismo equipo de diseño de la pregunta 1 ha pensado rediseñar su arquitectura del repertorio de instrucciones para contemplar la máquina R-M con una referencia a memoria (CPU B). A partir del análisis realizado, se contempla que sean 48 el número de operaciones a realizar incluyendo las aritméticas, lógicas, saltos, y referencias a memoria que permiten abordar las tareas de un procesador de propósito general. Dado que los programas que se ejecuten en el procesador se desarrollarán con lenguajes de alto nivel, se contempla un banco de 32 registros de 32 bits. Por último, el espacio de direcciones de acceso a memoria es de 4GB.

- a) Indica qué decisiones de manera **justificada** debería tomar el equipo de diseño, valorando las diferentes alternativas, en las siguientes componentes de la arquitectura:
 - i. Número de operandos de instrucciones ALU
 - ii. Modos de direccionamiento
 - iii. Codificación de los modos de direccionamiento
 - iv. Tipos de instrucciones en el repertorio
 - v. Formas de especificar el destino del salto y la condición de salto
- b) Determina los formatos de instrucciones incluyendo los campos de la instrucción y su tamaño según las decisiones tomadas en el apartado anterior.

1. Número de operandos de instrucciones ALU

Es preferible diseñar las instrucciones ALU con 3 operandos, ya que tenemos registros de sobra (32 son bastantes). Así, conseguimos una codificación más sencilla y una complejidad de instrucción menor, aunque, a diferencia de la opción con 2 operandos, perdemos rango de direccionamiento para el de la referencia a memoria.

2. Modos de direccionamiento

Como se trata de un procesador de propósito general, lo ideal es implementar los modos más utilizados: de registro, inmediato y de desplazamiento. Podemos, incluso, definir el modo indirecto -el siguiente más utilizado- como un caso particular de desplazamiento (cuando el mismo es 0).

3. Codificación de los modos de direccionamiento

Como tan sólo vamos a implementar unos pocos modos de direccionamiento, lo ideal es que la codificación sea de tipo fija y además con el código incluido en la operación (al necesitar pocos bits para codificar todas las 48 instrucciones, podemos aprovechar los que sobran para codificar los posibles modos de direccionamiento)

4. Tipos de instrucciones en el repertorio

A parte de las de tipo ALU, Load/Store y de Control, también es interesante incluir algunas para el sistema y también de punto flotante debido a que el procesador será de propósito general. Sin embargo, las de tipo decimal, para cadenas y sobre todo para gráficos pueden acarrear en la complejidad del procesador y, como precisamente el procesador no tiene un fin específico, es preferible no incluirlas e implementarlas por software.

5. Formas de especificar el destino del salto y la condición del salto

Si bien la que usa un código de condición puede provocar problemas de segmentación y la que tiene la comparación incluida en el salto puede hacer que se incremente el CPI, a lo mejor es conveniente implementar la que utiliza un registro para la condición debido a la cantidad que de la que disponemos.

b) Ni idea

Ejercicio 3 Junio 2015 AC

En el proceso de diseño de un computador se ha optado por una filosofía RISC. Indica qué decisiones tomarías, valorando las diferentes alternativas, en las siguientes componentes de la arquitectura:

- a) Número de operandos que se pueden direccionar en memoria en instrucciones ALU
 - b) Modos de direccionamiento de operandos
 - c) Tipo de instrucciones en el repertorio
 - d) Codificación de los modos de direccionamiento. Pon un ejemplo del formato de codificación elegido.
-
- a) Tenemos la opción de escoger tanto de 3 operandos (dos fuentes y un destino) como de 2 (un fuente y otro fuente-destino). Como en las arquitecturas RISC suele haber un gran número de registros, es mejor optar por la de 3 operandos ya que evitaremos la destrucción de un operando fuente (nos ahorraría una carga auxiliar del dato de una de las fuentes) sin que nos afecte mucho el hecho de que consumamos un registro extra para la operación.
 - b) Lo recomendable es tan solo implementar los modos más utilizados: de registro, inmediato y de desplazamiento. Los demás (indirecto, diferido, autoincremento-autodecremeno) los podemos simular o bien como caso particular de uno de los que tenemos o bien con instrucciones de más. Optando por escoger tan sólo unos pocos conseguimos facilitar su codificación a costa de un mayor RI.
 - c) Obligatoriamente debemos de incluir las esenciales: Aritmético-Lógicas, Carga-Almacenamiento y de Control. Como no se nos especifica el propósito del procesador, no sería necesario, aunque sí recomendable, incluir también instrucciones de Punto Flotante y, por si acaso, de Llamadas al Sistema. Otras como las Decimales, de Cadenas de Texto o Gráficas serían sólo necesarias si el procesador se fuese a utilizar en un ámbito muy específico.
 - d) Las arquitecturas RISC siempre eligen una codificación fija para aprovecharse de que las instrucciones tengan siempre la misma longitud (importante para la segmentación) incluyendo el modo de direccionamiento en el código de la operación. La razón por la cual eligen ésta y no la variable o híbrida es precisamente por la poca cantidad de instrucciones y modos de direccionamiento existentes. Un ejemplo de lo anteriormente explicado son las instrucciones tipo I (Inmediato) del RISC: 6 bits de código de operación, 5 bits para el registro destino, 5 bits para el registro fuente y 16 para el inmediato. Esta instrucción sirve, por ejemplo, para cargar un valor en memoria haciendo uso del modo de direccionamiento de desplazamiento (en el registro fuente aparece la referencia a memoria y el inmediato es utilizado como el desplazamiento en sí).

Ejercicio 4 junio 2015 AC

- a) Considerad un procesador no segmentado con una ruta de datos de 5 etapas de ejecución que funciona a 2GHz, en el que las operaciones ALU y salto requieren 4 ciclos de reloj y las de memoria 5. Suponed que las frecuencias relativas para estas operaciones son 45%, 25% y 30% respectivamente. Se quiere segmentar la máquina con 5 etapas y debido al sesgo del reloj y a los registros de segmentación se alarga el período de reloj en un 10%. El procesador utiliza una caché unificada para datos e instrucciones con un único puerto para el acceso a la memoria lo que provoca un riesgo estructural entre las etapas IF y MEM y por consiguiente una parada de 1 ciclo de reloj. Suponed que el CPI ideal del procesador segmentado ignorando el riesgo estructural es 1. ¿Cuál es la ganancia que se puede conseguir con la segmentación?

La ganancia se calcula de la siguiente manera:

$$\text{Aceleración}_{mejorada} = \frac{T \cdot CPU_{normal}}{T \cdot CPU_{segmentado}} = \frac{RI_{normal} \cdot CPI_{normal} \cdot CLK_{normal}}{RI_{segmentado} \cdot CPI_{segmentado} \cdot CLK_{segmentado}}$$

La mejora de segmentación no modifica el Recuento de Instrucciones (y, por tanto, son iguales):

$$\text{Aceleración}_{mejorada} = \frac{CPI_{normal} \cdot CLK_{normal}}{CPI_{segmentado} \cdot CLK_{segmentado}}$$

Podemos calcular el CPI del procesador sin segmentar con los datos que nos da el enunciado:

$$\begin{aligned} CPI_{normal} &= \%_{ALU} \cdot CPI_{ALU} + \%_{JMP} \cdot CPI_{JMP} + \%_{MEM} \cdot CPI_{MEM} = 0'45 \cdot 4 + 0'25 \cdot 4 + 0'30 \cdot 5 \\ &= 4'3 \end{aligned}$$

En el procesador segmentado, el porcentaje de uso sigue siendo el mismo (el RI no cambia), pero ahora el CPI nuevo es el ideal:

$$\begin{aligned} CPI_{segmentado} &= \%_{ALU} \cdot CPI_{ideal} + \%_{JMP} \cdot CPI_{ideal} + \%_{MEM} \cdot CPI_{ideal} \\ &= 0'45 \cdot 1 + 0'25 \cdot 1 + 0'30 \cdot 1 = 1 \end{aligned}$$

Sabemos que ahora el reloj del segmentado es un 10% más lento que el del normal:

$$CLK_{segmentado} = 1'1 \cdot CLK_{normal}$$

Conociendo todos los datos, ya podemos averiguar la ganancia:

$$\text{Aceleración}_{mejorada} = \frac{4'3 \cdot CLK_{normal}}{1 \cdot 1'1 \cdot CLK_{normal}} = \frac{4'3}{1'1} = 3'91$$

- b) Suponed ahora que además se consideran las detenciones por dependencia de datos de 1 ciclo de reloj que representan el 20% de todas las instrucciones ejecutadas y las detenciones por riesgo de control de 2 ciclos y que representan el 5% de todas las instrucciones ejecutadas. ¿Cuál es ahora el nuevo CPI? ¿En qué porcentaje se reduce la ganancia al considerar esta nueva situación?

Al CPI ideal le debemos de sumar las detenciones que causan los riesgos de la segmentación por la frecuencia con la que ocurren.

$$CPI_{segmentado} = 1 + 0'2 \cdot 1 + 0'05 \cdot 2 = 1'3$$

Ahora la ganancia es la siguiente:

$$Aceleración_{mejorada} = \frac{4'3 \cdot CLK_{normal}}{1'3 \cdot 1'1 CLK_{normal}} = \frac{4'3}{1'43} = 3'007$$

Por tanto, la ganancia se ve reducida en un:

$$\frac{3'91}{3'007} = 0'76 = -23\%$$

- c) Suponed que el siguiente código MIPS se ejecuta en la máquina segmentada:

addi \$3, \$0, 100

add \$4, \$0, \$0

Loop: lw \$5, 0(\$1)

add \$4, \$4, \$5

lw \$6, 0(\$2)

sub \$4, \$4, \$6

addi \$1, \$1, 4

addi \$2, \$2, 4

addi \$3, \$3, -1

bne \$3, \$0, Loop

Muestra el diagrama de temporización para una iteración del bucle suponiendo que no hay forwarding. Completa para ello la siguiente tabla y muestra todos los ciclos de paradas. Suponed que los saltos paran la segmentación solo durante un ciclo de reloj.

Antes de empezar, recordemos las restricciones:

- ⑩ Hay riesgo estructural entre IF y MEM
- ⑩ No habla de que no se pueda leer y escribir en registros a la vez (escribir en flanco de subida y leer en flanco de bajada), así que suponemos que se puede
- ⑩ No hay forwarding
- ⑩ Los saltos provocan una burbuja de tan sólo un ciclo

Instrucción	Ciclos de reloj																												
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26			
addi \$3, \$0, 100	IF	ID	EX	ME	WB																								
add \$4, \$0, \$0		IF	ID	EX	ME	WB																							
lw \$5, 0(\$1)			IF	ID	EX	ME	WB																						
add \$4, \$4, \$5				S	S	S	IF	ID	EX	ME	WB																		
lw \$6, 0(\$2)							IF	ID	EX	ME	WB																		
sub \$4, \$4, \$6								IF	S	S	ID	EX	ME	WB															
addi \$1, \$1, 4										IF	ID	EX	ME	WB															
addi \$2, \$2, 4											IF	ID	EX	ME	WB														
addi \$3, \$3, -1												S	S	S	IF	ID	EX	ME	WB										
bne \$3, \$0, Loop															IF	S	S	ID	EX	ME	WB								
lw \$5, 0(\$1)																S	S	S	ID	EX	ME	WB							
add \$4, \$4, \$5																	S	S	S	ID	EX	ME	WB						

Leyenda: amarillo → riesgo estructural, verde → dependencia de datos, azul → riesgo de control

1. (2,5 puntos) Sobre rendimiento

La empresa “3D Computer Vision processors” está diseñando el microprocesador 3D-RGBD-P especializado en visión, y software especializado para integrarlo en pequeñas cámaras RGB-D. El microprocesador está especializado en funcionalidades como filtros 3D y compresión/descompresión. Estas funcionalidades pueden ser ejecutadas siguiendo el modelo SIMD mediante GPUs integradas en el 3D-RGBD-P. Se han diseñado diferentes modelos del sistema 3D-RGBD-P que integran diferentes modelos de GPU (GPU 1,2,3,4) con diferentes prestaciones. Se ha realizado un estudio del software del sistema sobre el modelo 3D-RGBD-P-0 (que no integra GPU) utilizando benchmarks estándar y se ha determinado que el 90% del tiempo de ejecución del software se realiza tarea paralelizable mediante modelo SIMD. Tras analizar los cuatro modelos de GPU y sus posibilidades de paralelización, se concluye que las aceleraciones mejoradas de cada modelo de GPU respecto a la CPU integrada en el 3D-RGBD-P-0 son las siguientes (estas aceleraciones mejoradas se refieren a la parte paralelizable):

Modelo de 3D-RGBD-P	Tipo de GPU	Aceleración mejorada (solo lo paralelizable)
3D-RGBD-P-0	CPU (sin GPU)	1
3D-RGBD-P-1	GPU 1	100
3D-RGBD-P-2	GPU 2	500
3D-RGBD-P-3	GPU 3	1000
3D-RGBD-P-4	GPU 4	5000

Tras realizar varias pruebas del software con benchmarks estándar sobre el microprocesador más simple 3D-RGBD-P-0 que no integra ningún modelo de GPU se observa que el tiempo de ejecución (no paralelizable + paralelizable) es de 60s, lo que no resulta aceptable para los requerimientos de rendimiento esperados para este tipo de sistemas.

- a) (1 punto) Calcula los tiempos de ejecución del software (no paralelizable + paralelizable) en cada uno de los modelos del 3D-RGBD-P desde el 1 al 4.

		aceleración mejorada	aceleración global	no mejorable	Mejorable/ mejorado	no mejorable + mejorado
3D-RGBD-P-0	CPU (sin GPU)	1	1	6	54	60s
3D-RGBD-P-1	GPU 1	100	9,17	6	0,54	6,54s
3D-RGBD-P-2	GPU 2	500	9,82	6	0,108	6,108s
3D-RGBD-P-3	GPU 3	1000	9,91	6	0,054	6,054s
3D-RGBD-P-4	GPU 4	5000	9,98	6	0,0108	6,0108s

- b) (1,5 punto) Una empresa de instrumentación médica está valorando integrar estos microprocesadores en sus sistemas. Para ello demanda que los benchmarks estándar se ejecuten en 5s o menos.

- a) ¿Alguna de las versiones del 3D-RGBD-P cumple los requerimientos? Justifica la respuesta (0,5 puntos)

Ninguna. El modelo más rápido (3D-RGBD-P-4) tarda 6,0108s, muy por encima de los 5s requeridos.

- b) ¿Sería posible alcanzar el requerimiento de 5s manteniendo el 90% del software paralelizable, mejorando la aceleración mejorada de la GPU? Justifica la respuesta (0,5 puntos)

Imposible. Cuando la aceleración mejorada tiende a infinito el tiempo de la parte paralelizable (mejorable) tiende a cero, pero la parte no paralelizable (no mejorable) se mantiene en 6s, muy por encima de los 5s requeridos.

- c) ¿Cuánto habría que aumentar el porcentaje de paralelización sobre el 90% con la GPU4 para cumplir el requerimiento de 5s? (**0,5 puntos**)

La aceleración global necesaria para alcanzar el requerimiento de 5s es:

$$A_g = \frac{60}{5} = 12$$

Para conseguir la aceleración global 12 con una aceleración mejorada 5000 (correspondiente al modelo de la GPU4) la fracción mejorada necesaria sería:

$$A_g = 12 = \frac{1}{(1 - f_{m\ requierida}) + \frac{f_{m\ software}}{5000}} \rightarrow f_{m\ software} = 0,91685$$

El incremento en el porcentaje de paralelización (fracción mejorada) sería

$$\Delta_{fm} = |f_{m\ requerida} - f_{m\ inicial}| = |0,91685 - 0,9| = 0,01685 = \mathbf{1,69\%}$$