Tecnológico de Costa Rica

Ingeniería en Computadores

Arquitectura de Computadores I

Tarea 2: Práctica para examen I

Estudiante:
Malcolm Davis

Profesor:
Fabián Zamora

22 de septiembre de 2017

Resumen

Este trabajo se desarrolló para reforzar los conceptos vistos en clases sobre los temas que se evaluarán en el primer examen del curso. Consta de 10 ejercicios tanto teóricos como prácticos.

1. Pregunta 1

Mencione las categorías en las que dividió Flynn los procesadores. Presente un ejemplo de al menos dos de ellas.
Flynn hizo esta clasificación tomando en cuenta la cantidad de instrucciones concurrentes y el flujo de datos.[1]

1.1. SISD

Por sus siglas, y traducido al español, Instrucción Única y flujo de Datos Único. Como lo indica su nombre sería una computadora secuencial que no usa paralelismo.

1.2. SIMD

Por sus siglas, y traducido al español, Instrucción Única y flujo de Datos Múltiples. Esta sería una computadora que le aplique la misma instrucción a diferentes flujos de datos paralelamente, ejemplos de esto podría ser el procesamiento que se hace en la GPU.

1.3. MISD

Por sus siglas, y traducido al español, Instrucciones Múltiples y flujo de Datos Único. Está categoría no es muy utilizada comúnmente, le aplica varias instrucciones a un set de datos. Las literaturas difieren sobre si existen o no implementaciones de esta categoría, pero en la literatura recomendada del curso afirma que no las hay.

1.4. MIMD

Por sus siglas, y traducido al español, Instrucciones Múltiples y flujo de Datos Múltiple. Cada procesador recolecta las instrucciones a aplicar a su set de datos. Un ejemplo de esto es un cluster o súper computadora.

2. Pregunta 2

Asuma que se hace un cambio a un computador que mejora cierto modo de ejecución (A) en un factor de 10. Este modo A es utilizado el 50 % del tiempo (este 50 % es del tiempo cuando se utiliza el modo y mejorado). Recuerde que la ley de Amdahl depende de la fracción del tiempo de ejecución SIN MEJORAR que puede ser mejorado. De forma que no se puede utilizar este 50 % para calcular el speedup utilizando la ley de Amdahl. Sabiendo que:

$$Speedup_{overal} = \frac{Executiontime_{old}}{Executiontime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

2.1. ¿Cuál es el speedup obtenido utilizando el modo mejorado?

Suponiendo que el computador mejorado dura un tiempo $2x$, y tomando en cuenta que el porcentaje del tiempo para el modo de ejecución A mejorado es de 50 %. El tiempo de ejecución de A sería de x y el tiempo de ejecución del resto que no se pudo mejorar sería igual de x. Al ser una mejora de un factor de 10, significa que sólo esa parte se ejecuta 10 veces más rápido, por lo tanto: El tiempo de ejecución antiguo de A sería $Aexcecutiontime_{old} = 10x$. Por lo tanto se cumple que:

$$Fraction_{enhanced} = \frac{Aexecutiontime_{old}}{Executiontime_{old}} = \frac{10x}{10x + x} = 0{,}91$$

Y por último se tiene que:

$$Speedup_{overal} = \frac{1}{(1 - \frac{10}{11}) + \frac{\frac{10}{11}}{10}} = 5{,}5$$

2.2. ¿Qué porcentaje del tiempo de ejecución original se ha mejorado?

Como se calculó para la pregunta anterior:

$$Fraction_{enhanced} = \frac{Aexecutiontime_{old}}{Executiontime_{old}} = \frac{10x}{10x + x} = 0{,}91$$

Por lo tanto se mejoró un 91 % del tiempo original.

3. Pregunta 3

Al hacer cambios para optimizar parte de un computador, en ocasiones, al mejorar una característica se empeora otra. Por ejemplo, al mejorar una unidad de punto flotante compleja, el área agregada puede influir negativamente en el tiempo de acceso a la memoria cache de primer nivel (L1), agregando un ciclo extra. La ley de Amdahl, como tal, no toma en cuenta estas situaciones.

3.1. Si la nueva unidad de punto flotante mejora las operaciones en 2x, y dichas operaciones toman 20 % del tiempo de ejecución el sistema original ¿Cuál es la mejora general del sistema?

Similar a lo visto en la pregunta anterior, se tiene que el $Speedup_{enhanced} = 2$ y que el $Fraction_{enhanced} = 0{,}2$. Usando la ley de Amdahl se obtiene que:

$$Speedup_{overal} = \frac{1}{(1 - 0{,}2) + \frac{0{,}2}{2}} = 1{,}1$$

3.2. Ahora asuma que la nueva unidad de punto flotante (descrita en a.) empeora los accesos a memoria cache L1, resultando en un empeoramiento de los mismos de 1.5x (mejora de 2/3x). Si los accesos a caché L1 consumen 10 % del tiempo de ejecución del sistema original ¿Cuál es ahora la mejora general en el sistema?

Para poder resolver este problema, se debe de tomar en cuenta que ambas modificaciones ocurren al mismo tiempo sobre el sistema inicial, por lo tanto, hay que calcular un $Speedup_{overall}$ que incluya ambas. Al igual que en los otros, se calcula el $Fraction_{enhaced}$ y el $Speedup_{enhaced}$ de cada modificación, tal que $Fraction_{enhaced}Fp = 0{,}2$, $Speedup_{enhaced}Fp = 2$, $Fraction_{enhaced}L1p = 0{,}1$ y $Speedup_{enhaced}L1 = 2/3$. Por último, se deduce:

$$Speedup_{overall} = \frac{1}{(1 - Frac_{enhacd}Fp - Frac_{enhac}L1) + \frac{Frac_{enhac}Fp}{Speedup_{enhac}Fp} + \frac{Frac_{enhac}L1}{Speedup_{enhac}L1}}$$

$$= \frac{1}{(1 - 0{,}2 - 0{,}1) + \frac{0{,}2}{2} + \frac{0{,}1}{\frac{2}{3}}} = 1{,}05$$

4. Pregunta 4

Su compañía acaba de comprar un nuevo procesador Intel Core i5, de doble núcleo, y se le ha asignado a usted optimizar el software para ese procesador. Usted ejecutará dos aplicaciones en el procesador, pero los recursos requeridos no son iguales. La primera aplicación requiere 80 % de los recursos y la otra el 20 % restante. Asuma que cuando paraleliza una porción del programa, la mejora de esa porción es de 2.

4.1. Dado que el 40 % de la primera aplicación es paralelizable ¿Qué mejora obtendría en el sistema si solo esta aplicación se ejecuta?

Partiendo de la suposición que sólo la primera aplicación se ejecuta en el sistema (la aplicación representa el 100 % del tiempo del sistema); la mejora $Speedup_{enahaced}$ es de 2, y la fracción de tiempo o $Fraction_{enhaced}$ es 0.4. A partir de la ley de Amdahl se construye:

$$Speedup_{overal} = \frac{1}{(1 - 0{,}4) + \frac{0{,}4}{2}} = 1{,}25$$

4.2. Dado que el 99 % de la segunda aplicación es paralelizable ¿Qué mejora obtendría en el sistema si solo esta aplicación se ejecuta?

Partiendo de la misma suposición que para el punto anterior(la aplicación representa el 100 % del tiempo del sistema); la mejora $Speedup_{enahaced}$ es de 2, y la fracción de tiempo o $Fraction_{enhaced}$ es 0.99. A partir de la ley de Amdahl se construye:

$$Speedup_{overal} = \frac{1}{(1 - 0{,}99) + \frac{0{,}99}{2}} = 1{,}98$$

4.3. Dado que el 99 % de la segunda aplicación es paralelizable ¿Qué mejora observaría el sistema general (ambas aplicaciones se ejecutan)?

Tomando en cuenta los datos iniciales brindados, donde aseguran que la aplicación 1 utiliza el 80 % de los recursos, mientras la aplicación 2 utiliza el 20 % de los recursos. Y además la nueva información brindada, que establece que de ese 80 % de los recursos que utiliza la aplicación, sólo se puede paralelizar un 40 % de la aplicación lo que significa que del sistema original para la aplicación 1 se puede paralelizar sólo un $Fraction_{enhaced}1 = 80\,\% * 0{,}4 = 32\,\%$ y del 20 % de recursos restantes, se puede paralelizar un 99 % de la aplicación lo que significa que del sistema original se tiene un $Fraction_{enhaced}1 = 20\,\% * 0{,}99 = 19{,}8\,\%$ paralelizable. Y para ambos se tiene un $Speedup_{enhaced} = 2$. Por lo tanto, usando Amdahl:

$$Speedup_{Overall} = \frac{1}{(1 - 0{,}32 - 0{,}19) + \frac{0{,}32+0{,}19}{2}} = 1{,}34$$

5. Pregunta 5

Al paralelizar una aplicación, la mejora ideal es equivalente al número de procesadores. Esto es limitado por dos aspectos: porcentaje de la aplicación paralelizable y costo de comunicación. La ley de Amdahl toma en cuenta el primer factor, pero no el segundo.

5.1. ¿Cuál es la mejora con 8 procesadores si el 80 % de la aplicación es paralelizable, ignorando la comunicación?

Tomando en cuenta que al utilizar 8 procesadores para paralelizar la aplicación se obtiene una mejora del tiempo de $Speedup_{enhaced} = 8$ y se tiene que la fracción paralelizable es $Fraction_{enhaced} = 0{,}8$.

$$Speedup_{overal} = \frac{1}{(1 - 0{,}8) + \frac{0{,}8}{8}} = 3{,}3$$

5.2. ¿Cuál es la mejora con 8 procesadores si, para cada procesador, el overhead de comunicación es de 0,5 % del tiempo de ejecución original?

Similar al punto anterior, se tiene un $Speedup_{enhacement}1 = 8$ al utilizar 8 procesadores, y al ser un 80 % de la aplicación paralelizable entonces existe un $Fraction_{enhaced}1 = 0{,}8$ pero también al tener que tomar en cuenta la parte que se consume en comunicación, la cual tiene un $Speedup_{enhacement}2 = \frac{1}{2}$ o dura el doble de tiempo(esta fue una suposición hecha con la solución que brindó el profesor, ya que en ningún lado del problema dice cuanto empeora), y un $Fraction_{enhaced} = 0{,}005$. Por último, con Amdahl se encuentra el $Speedup_{overall}$:

$$Speedup_{overall} = \frac{1}{(1 - 0{,}8 - 0{,}005 * 8) + \frac{0{,}8}{8} + \frac{0{,}005}{\frac{1}{2}} * 8} = 2{,}94$$

6. Pregunta 6

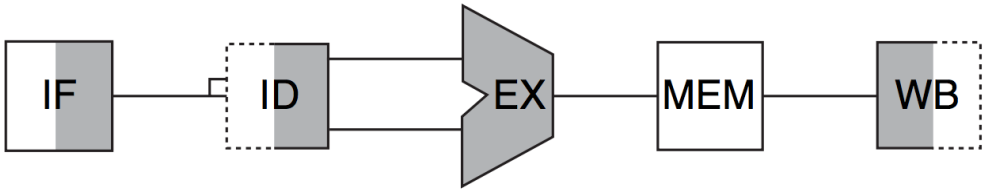

Figura 1: Pipeline de 5 Etapas

Para las secuencias de instrucciones de la figura 2:

6.1. Encuentre todas las dependencias de datos.

6.1.1. Dependencias Set A

1. Existe una dependencia de nombre con el registro 2, entre la instrucción lw y el primer add.

2. Existe una dependencia de nombre con el registro 1, entre la instrucción lw y el segundo add.

	Instruction sequence

a.	`lw  $1,40($2)`
	`add $2,$3,$3`
	`add $1,$1,$2`
	`sw  $1,20($2)`

b.	`add $1,$2,$3`
	`sw  $2,0($1)`
	`lw  $1,4($2)`
	`add $2,$2,$1`

Figura 2: Instrucciones a Analizar

3. Existe una dependencia de datos con el registro 1, entre la instrucción lw y el segundo add.

4. Existe una dependencia de datos con el registro 2, entre la primera y la segunda instrucción add..

5. Existe una dependencia de datos con el registro 1, entre la instrucción lw y la instrucción sw.

6. Existe una dependencia de datos con el registro 1, entre la segunda instrucción add y la instrucción sw.

7. Existe una dependencia de datos con el registro 2, entre la primera instrucción add y la instrucción sw.

6.1.2. Dependencias Set B

1. Existe una dependencia de datos con el registro 1, entre la primera instrucción add y la instrucción sw.

2. Existe una dependencia de nombre con el registro 1, entre la primera instrucción add y la instrucción lw.

3. Existe una dependencia de nombre con el registro 1, entre la instrucción sw y la instrucción lw.

4. Existe una dependencia de nombre con el registro 2, entre la primera instrucción add y la segunda instrucción add.

5. Existe una dependencia de nombre con el registro 2, entre la instrucción sw y la segunda instrucción add.

6. Existe una dependencia de nombre con el registro 2, entre la instrucción lw y la segunda instrucción add.

7. Existe una dependencia datos con el registro 1, entre la instrucción lw y la segunda instrucción add.

Lo cual se puede traducir en función de riesgos a:

6.1.3. Riesgos Set A

1. Existe un riesgo de **W**rite **A**fter **R**ead con el registro 2, entre la instrucción lw y el primer add.

2. Existe un riesgo de **W**rite **A**fter **W**rite con el registro 1, entre la instrucción lw y el segundo add.

3. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y el segundo add.

4. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 2, entre la primera y la segunda instrucción add..

5. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la instrucción sw.

6. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la segunda instrucción add y la instrucción sw.

7. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 2, entre la primera instrucción add y la instrucción sw.

6.1.4. Riesgos Set B

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la primera instrucción add y la instrucción sw.

2. Existe un riesgo de **W**rite **A**fter **W**rite con el registro 1, entre la primera instrucción add y la instrucción lw.

3. Existe un riesgo de de **W**rite **A**fter **R**ead con el registro 1, entre la instrucción sw y la instrucción lw.

4. Existe un riesgo de **W**rite **A**fter **R**ead con el registro 2, entre la primera instrucción add y la segunda instrucción add.

5. Existe un riesgo de **W**rite **A**fter **R**ead con el registro 2, entre la instrucción sw y la segunda instrucción add.

6. Existe un riesgo de **W**rite **A**fter **R**ead con el registro 2, entre la instrucción lw y la segunda instrucción add.

7. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la segunda instrucción add.

6.2. Encuentre todos los riesgos para un pipeline de 5 etapas si no se utiliza adelantamiento.

Como se vio en lecciones, los riesgos reales para un pipeline de 5 etapas son los relacionados con las dependencias reales y serían riesgos **R**ead **A**fter **W**rite, ya que los tipos de dependencia de nombre, no presentan riesgo porque no hay un flujo real de datos(se usa el mismo registro pero la información que se almacena, no se usa en la instrucción con la cual se tiene la dependencia). Por lo tanto, los riesgos reales para la micro-arquitectura analizada serían:

6.2.1. Riesgos Reales Set A

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y el segundo add.

2. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 2, entre la primera y la segunda instrucción add..

3. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la instrucción sw.

4. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la segunda instrucción add y la instrucción sw.

5. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 2, entre la primera instrucción add y la instrucción sw.

6.2.2. Riesgos Reales Set B

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la primera instrucción add y la instrucción sw.

2. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la segunda instrucción add.

6.3. Encuentre todos los riesgos para un pipeline de 5 etapas utilizando adelantamiento.

Al utilizar adelantamiento, el resultado de la etapa de ejecución se hace disponible en el siguiente ciclo a la ALU para la siguiente instrucción, por esto se corrigen algunos de los riesgos. Los riesgos que quedan son:

6.3.1. Riesgos Reales Adelantamiento Set A

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y el segundo add.

2. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la instrucción sw.

6.3.2. Riesgos Reales Adelantamiento Set B

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la segunda instrucción add.

6.4. Para reducir el tiempo del ciclo de reloj, se considera separar la etapa de MEM en 2 etapas. Repita los puntos 6.2 y 6.3 para este pipeline con 6 etapas.

Encuentre todos los riesgos para un pipeline de 5 etapas si no se utiliza adelantamiento.

6.4.1. Riesgos Reales Set A

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y el segundo add.
2. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 2, entre la primera y la segunda instrucción add.
3. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la instrucción sw.
4. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la segunda instrucción add y la instrucción sw.
5. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 2, entre la primera instrucción add y la instrucción sw.

6.4.2. Riesgos Reales Set B

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la primera instrucción add y la instrucción sw.
2. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la segunda instrucción add.

Encuentre todos los riesgos para un pipeline de 5 etapas utilizando adelantamiento.

6.4.3. Riesgos Reales Adelantamiento Set A

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y el segundo add.
2. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la instrucción sw.

6.4.4. Riesgos Reales Adelantamiento Set B

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la instrucción lw y la segunda instrucción add.

Asuma que antes de ejecutar las instrucciones anteriores se tienen los valores de la figura 3 para los registros $0a$**3:**

	$0	$1	$2	$3
a.	0	1	31	1000
b.	0	–2	63	2500

Figura 3: Valores iniciales de los registros

6.5. ¿Cuáles son los valores de los registros al final de cada secuencia de instrucciones? ¿Cuál valor es el primero en ser adelantado y cuál valor sobrescribe?

6.5.1. Set A

Los valores luego de la ejecución serían $0=0, $1=32,$2=2000,$3=1000. El primer valor en ser adelantado es el de $2 = 2000 que sobre escribe el valor de 31 original.

6.5.2. Set B

Los valores luego de la ejecución serían $0=0, $1=0,$2=2626,$3=2500. El primer valor en ser adelantado es el de $1 = 2563 que sobre escribe el valor de -2 original.

6.6. Si usted por error olvidó implementar el hardware para adelantamiento, ¿cuáles son los valores de los registros al final de cada secuencia de instrucciones?

6.6.1. Set A

Los valores luego de la ejecución serían $0=0, $1=2000+MEM[40+32],$2=2000,$3=1000. El primer valor en ser adelantado es el de $2 = 2000 que sobre escribe el valor de 31 original.

6.6.2. Set B

Los valores luego de la ejecución serían $0=0, $1=MEM[2563+4],$2=MEM[2563+4]+2563,$3=2
El primer valor en ser adelantado es el de $1 = 2563 que sobre escribe el valor de -2 original.

6.6.3. Set B

6.7. Agregue nops a la secuencia de instrucciones para evitar los problemas vistos en el punto f.

6.7.1. Set A

Para resolver estos problemas, se debe agregar 2 nops entre el primer y el segundo add y otros dos nops entre el segundo add y el sw.

6.7.2. Set B

Para resolver estos problemas, se debe agregar 2 nops entre el primer add y el sw y otros dos nops entre el lw y el segundo add.

7. Pregunta 7

Con base en el código fuente de la figura 4:

```
int n=1024;
for (x=0, x<n, x++) {
    if (x<n/2){
        a[x]=1;
    }
    else{
        a[x]=0;
    }
}
```

Figura 4: Código a Analizar

7.1. Convierta el código mostrado a ensamblador (MIPS)

```
1     .data
2 a:    .space 4036
3 n:    .word 1024
4 init  li $1, 0    ;x
5     lw $2, n    ;load a size
6     la $3, a    ;load a address
7     div $4, $2, 2 ;save n/2 in a register
8 ciclo slt $5, $1, $4  ;set a register to 1 if x<n/2
9     sw $3, 0($5)  ;a[x]=
10    addi $1, $1, 1  ;x++
11    addi $3, $3, 4  ;next array item
12    beq $1, $2, 4
13    j ciclo
```

7.2. Identifique los posibles riesgos en el código de ensamblador para una arquitectura con un pipeline de 5 etapas sin hardware replicado.

7.2.1. Riesgos Reales Set A

1. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 2, entre la instrucción lw y la instrucción div.

2. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 4, entre la instrucción div y la instrucción slt.

3. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 5, entre la instrucción slt y la instrucción sw.

4. Existe un riesgo **REAL** de **R**ead **A**fter **W**rite con el registro 1, entre la primera instrucción addi y la instrucción beq.

7.3. Determine el número de bloques básicos que posee el código en ensamblador, así como el contenido en instrucciones de cada bloque.

El código convertido en ensamblador tiene 2 bloques, en el **init** hay 4 instrucciones, y en el **ciclo** hay 6 instrucciones.

7.4. Si las instrucciones, en general, toman 5 ciclos de reloj en ejecutarse y considerando que cada bloque básico puede ejecutarse en paralelo mediante un pipeline balanceado (no considere dependencias ni riesgos, asuma que la predicción de saltos funciona en un 100 % de los casos) ¿cuál sería el WCET (en ciclos de reloj) para el código anterior?

Tomando en cuenta que las primeras 4 instrucciones del primer bloque no se repiten, y que las 6 instrucciones del segundo bloque se repiten n veces, y que en este caso $n = 1024$. Se tiene una cantidad total de instrucciones de 6148. Por lo tanto, el WCET estaría dado por:

$$WCTE = Instruction_{count} - 1 + CPI = 6148 - 1 + 5 = 6152$$

7.5. Considere ahora en la ejecución del código los posibles riesgos encontrados en el punto 7.2) ¿Cuántos stalls tendrían que agregarse para evitar dichos riesgos? ¿Cuál sería el nuevo tiempo de ejecución (en ciclos de reloj) del programa?

1. Para el primer riesgo, se agregan 2 stalls entre lw y div.
2. Para el segundo riesgo, se agregan 3 stalls entre div y slt.
3. Para el tercer riesgo, se agregan 3 stalls entre slt y sw.
4. Para el último riesgo existente, se agregan 2 stalls entre addi y beq.

Tomando en cuenta lo anterior, se necesitarían 10 stalls para evitar los riesgos. Y ya que de esos stalls, sólo los últimos 5 están dentro del ciclo, la nueva $Instruction_{count}$ estaría dada por:

$$Instruction_{count} = initInstructions_{count} + initStalls + (cicleInstructions_{count} + cicleStalls) * n$$

$$= 4 + 5 + (6 + 5) * 1024 = 11273$$

8. Pregunta 8

Con base en el código de la figura 5:

```
_init:
    LW   R1, A
    ADD  R3, R1, R2
    SLL  R1, R2, 3
    ADD  R1, R1, R2
    BEQ  R2, T0, ext
    SUB  R1, R2, R3
    SW   R2, A

_ext:
    done
```

Figura 5: Código a Analizar

8.1. Mencione todas las dependencias (datos, nombre, control) existentes en el mismo para una arquitectura con ejecución en orden y una fuera de orden.

8.1.1. Dependencias Arquitectura En Orden

1. Existe una dependencia de datos con el registro 1, entre la instrucción lw y el primer add.
2. Existe una dependencia de nombre con el registro 1, entre la instrucción lw y la instrucción sll.
3. Existe una dependencia de nombre con el registro 1, entre la primera instrucción add y la instrucción sll.
4. Existen dos dependencia de nombre con el registro 1, entre la instrucción lw y la segunda instrucción add.
5. Existe una dependencia de nombre con el registro 1, entre la primera y la segunda instrucción add.

6. Existe una dependencia de nombre con el registro 1, entre la instrucción sll y la segunda instrucción add.

7. Existe una dependencia de datos con el registro 1, entre la instrucción sll y la segunda instrucción add.

8. Existe una dependencia de nombre con el registro 1, entre la instrucción lw y la instrucción sub.

9. Existe una dependencia de nombre con el registro 1, entre la primera instrucción add y la instrucción sub.

10. Existe una dependencia de nombre con el registro 1, entre la instrucción sll y la instrucción sub.

11. Existen dos dependencia de datos con el registro 1, entre la segunda instrucción add y la instrucción sub.

12. Existe una dependencia de datos con el registro 1, entre la instrucción sll y la segunda instrucción add.

8.1.2. Dependencias Arquitectura Fuera de Orden

Las dependencias para una arquitectura fuera de orden son las mismas que para una arquitectura en orden, la diferencia es que en una arquitectura fuera de orden las dependencias de nombre y control puede presentar algún riesgo mientras que en las en orden no.

8.2. Para una arquitectura con un pipeline de 5 etapas y una posible ejecución fuera de orden determine cuáles riesgos se podrían presentar y cómo podrían solucionarse.

Tomando en cuenta una posible ejecución fuera de orden, todas las dependencias anteriores se convierten en riesgos, eliminando la dependencia 8.

9. Pregunta 9

La disponibilidad es la consideración más importante a la hora de diseñar servidores, seguido por la escalabilidad y el flujo de instrucciones.

9.1. Se tiene un procesador con un promedio de 100 fallas por año. ¿Cuál es el tiempo medio para falla (MTTF), en días, del sistema?

Tomando en cuenta el promedio de fallas brindado y suponiendo que no es un año bisiesto, se calcula el MTTF como:

$$MTTF = \frac{1}{100} * \frac{year}{fail} * \frac{365}{1} * \frac{days}{year} = 3{,}65 \frac{day}{fail}$$

Esto quiere decir que el MTTF de este procesador es 3.65 días, o que en promedio cada 3.65 días falla.

9.2. Si toma un día para tener el sistema funcionando de nuevo, ¿cuál es la disponibilidad del sistema?

Utilizando la formula brindada en clase:

$$ModuleAvailability = \frac{MTTF}{MTTF + MTTR} = \frac{3{,}65}{3{,}65 + 1} = 0{,}78$$

O que el sistema está disponible un 78 % del tiempo.

9.3. Suponga que el gobierno, para recortar gastos, va a construir un supercomputador de componentes económicos, en lugar de componentes más caros y confiables. ¿Cuál es el MTTF para el sistema con 1000 de los procesadores mencionados arriba? Asuma que los procesadores fallan simultáneamente.

Si todos los componentes fallan al mismo tiempo, entonces se deduce que:

$$MTTF_{system} = MTTF_{component} = 3{,}65 \frac{day}{fail}$$

10. Pregunta 10

En granjas de servidores usadas por Amazon o eBay, una sola falla no causa que el sistema entero falle. En su lugar, causa la reducción del número de peticiones que pueden ser atendidas simultáneamente.

10.1. Si una compañía tiene 10000 computadores, cada uno con un MTTF de 35 días, y experimenta una falla catastrófica solamente si 1/3 de los computadores fallan, ¿cuál es el MTTF del sistema?

Para que exista una falla del sistema 3333 computadoras tienen que fallar, y conociendo que el MTTF del sistema es el inverso de la suma del FIT de sus componentes, se puede calcular MTTF como:

$$MTTF_{system} = (FIT_{component} * component_count)^{-1} = (\frac{1}{35} * 3333)^{-1} = 0{,}01$$

10.2. Si cuesta $1000 extra, por computador, doblar el MTTF, ¿sería esta una buena decisión de negocio? Justifique su respuesta.

Para esta respuesta habría que tomar en cuenta muchos factores, como por ejemplo que tan necesario sería doblar el MTTF del sistema para esta compañía, si es critico mantener el sistema disponible la mayor cantidad de tiempo, entonces sería una buena opción a considerar, pero si en cambio no es tan importante, una inversión de $10000000 no sería una necesidad a corto plazo. Así que dependiendo del tipo de negocio la respuesta sería diferente.

Referencias

[1] J. HENNESSY, *COMPUTER ARCHITECTURE*, 5th ed. [S.l.]: MORGAN KAUFMANN, 2017, p. 39.