Universidad Católica San Pablo | Departamento de Ciencia de la Computación

Examen Final de Arquitectura de Computadores 2019-01

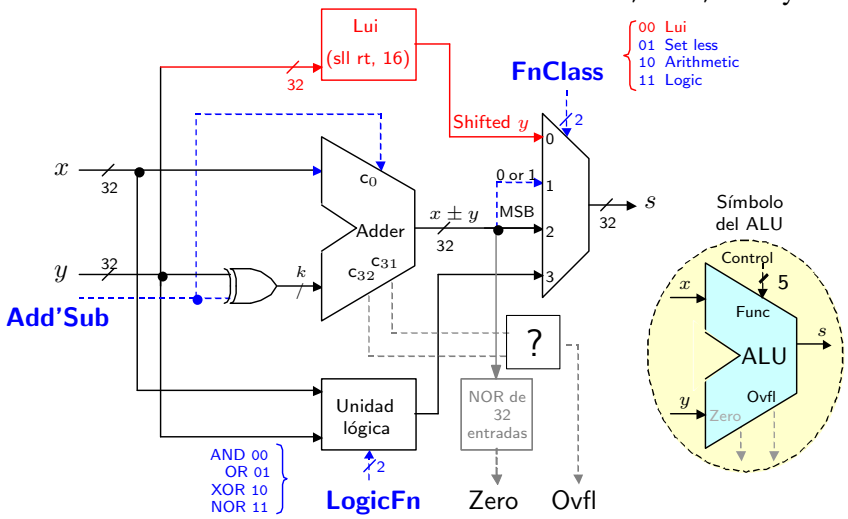
Escuela Profesional de Ciencia de la Computación: CCOMP3-2

02 de julio 2019

ARCHITECTURE IN SINGLE CYCLE

Pregunta 1: **[4pt, Habilidades: b2]**

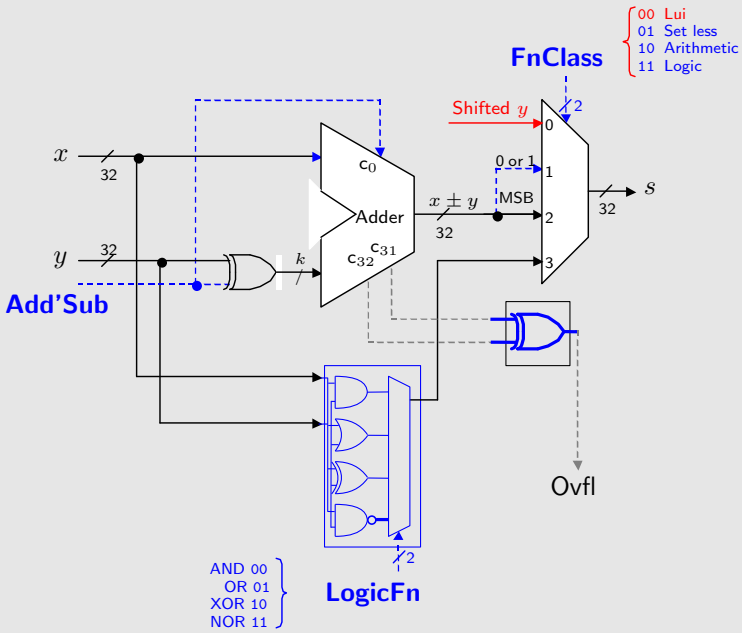
Sea la unidad aritmético-lógica de la figura, la cual ha sido adecuada para implementar la instrucción [`lui rt, imm`] en vez de las instrucciones de corrimiento estudiadas **`srl`**, **`sll`**, **`sra`** y sus versiones variable **`srlv`**, **`sllv`**, **`srav`**.



Haga el diseño lógico (diseño de puertas lógicas) faltante de:

a) El bloque que implementa la detección de *overflow* con salida Ovfl

b) El bloque que implementa las instrucciones lógicas **`and`**, **`or`**, **`xor`**, **`nor`**



ARCHITECTURE IN MULTICYCLE

Pregunta 2: **[4pt, Habilidades: b2]**

Sea una máquina de ejecución de instrucciones denominada **MicroMIPS++** que soporta más instrucciones que las estudiadas en clase, cuyas instrucciones soportadas tipo R tienen las siguientes variantes y porcentajes de ejecución:

a) Instrucciones R `add`, `sub`, `xor`, `and`, `or`, ... del MicroMIPS que gastan 4 ciclos de reloj, y se usan al 60%

b) Instrucciones R de corrimiento `srl`, `sll`, `sra`, `srlv`, `sllv`, `srav`, con 5 ciclos de reloj y uso al 20%

c) Instrucciones R de multiplicación entera `mul`, `multu` que gastan 8 ciclos de reloj y se usan al 15%

d) Instrucciones R de división `div`, `divu` que gastan 10 ciclos de reloj y se usan al 5%.

Considerando las siguientes estadísticas de ejecución para ciclos de reloj de 0.8ns.

Aritméticas (R)	50%	varios ciclos
Lectura en memoria	23%	5 ciclos
Guardar en memoria	10%	4 ciclos
Bifurcaciones	15%	3 ciclos
Saltos incondicionales	2%	3 ciclos

i) Analice el rendimiento si la arquitectura es de unico ciclo, determinando la cantidad de IPS (Instrucciones por segundo) para la máxima frecuencia de reloj posible.

ii) Calcule el rendimiento en IPS para una arquitectura multiciclo y determine la diferencia de rendimiento con respecto al ciclo único

i) Analice el rendimiento si la arquitectura es de unico ciclo, determinando la cantidad de IPS (Instrucciones por segundo) para la máxima frecuencia de reloj posible.

Debemos determinar la máxima frecuencia de reloj posible, para esto observemos el tipo de instrucción que usa más ciclos:, entre las *R* y las demás se tiene la de división que usa 10 ciclos de 0.8ns, haciendo 8ns de ejecución, por lo tanto el reloj puede ser a lo máximo 125MHz, lo cual nos permite calcular el IPS (Instrucciones por segundo) que es dado por

$$IPS_{single} = \frac{f}{CPI} = \frac{125MHz}{1c/ins} = 125MIPS$$

ii) Calcule el rendimiento en IPS para una arquitectura multiciclo y determine la diferencia de rendimiento con respecto al ciclo único

- Como las instruccionnes en multiciclo se ejecutan por partes, se puede colocar un reloj con el ciclo de reloj de 0.8ns que haría una frecuencia de 1.25GHz, luego para obtener el rendimiento en IPS es necesario calcular el CPI medio de la arquitectura, para esto usamos las estadísticas especificadas:

$$\begin{aligned} CPI &= 0.5 \times (4 \times 0.6 + 5 \times 0.2 + 8 \times 0.15 + 10 \times 0.05) + 5 \times 0.23 + 4 \times 0.10 + 3 \times 0.15 + 3 \times 0.02 \\ &= 0.5 \times (5.1) + 1.15 + 0.4 + 0.45 + 0.06 \\ CPI &= 4.61 \end{aligned}$$

- Luego el IPS sería:

$$IPS_{multi} = \frac{f}{CPI} = \frac{1.25GHz}{4.61c/ins} = 271.15MIPS$$

- Calcular la diferencia sería usar una expresión como:

$$Dif\% = \frac{CPI_{multi} - CPI_{single}}{CPI_{single}}\% \quad (1)$$

Donde reemplazando valores se obtiene:

$$Dif\% = \frac{271.15MIPS - 125MIPS}{125MIPS}\% \tag{2}$$

$$Dif\% = 116.92\% \tag{3}$$

ARCHITECTURE IN PIPELINING: BRANCH PREDICTION

Pregunta 3: **[4pt, Habilidades: b2,i3]**

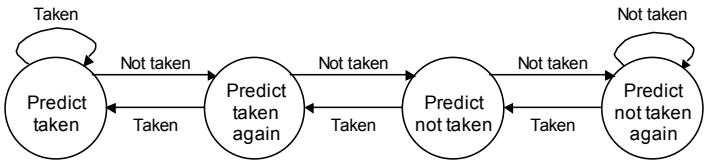
Sea el siguiente trecho que construye y almacena en memoria la serie de fibonacci de n términos empezando por $1,1,2,\ldots$ a ser ejecutado en una arquitectura en *pipeline* de 5 etapas:

```
fib:addi $t0,$zero,1
      sw $t0, 0($a1)       # mem[($a1)]   = 1
      sw $t0, 4($a1)       # mem[($a1)+4] = 1
    addi $a0, $a0,10       # n = 10
loop:                      # for ($a0)=10, ($a0)!=0, ($a0)--
      lw $t0, 0($a1)
      lw $t1, 4($a1)
     add $t0, $t0, $t1
      sw $t0, 8($a1)       # mem[($a1)+8]=mem[($a1)]+mem[($a1)+4]
    addi $a1, $a1, 4       # ($a1)+= 4
    addi $a0, $a0, -1      # ($a0)+=-1
     bne $a0, $zero,loop   # if ($a0)!= 0 goto loop
        endFib: ...
```

a) Detecte las dependencias de datos tipo I y II, y dependencias de control

b) Cuántos ciclos de reloj se utilizarán en la ejecución de ese código si se usa las siguientes capacidades en el *pipeline*:

- *Data Forwarding*, posibilidad de reordenaciones
- Previsión de Bifurcacion de 2bits que está en el estado *Predict not taken again* al inicio de este loop. Considere cada equivocación de la prediccion de bifurcación penalizada con dos ciclos de reloj.



c) **[bonificación extra 1pt]** Dé alguna sugerencia de cambio (armado del código, arquitectura de ejecución u otros) para que se ejecute este proceso en menos ciclos de reloj.

a) Las dependencias se encuentran de la siguiente forma:

- Dependencia datos tipo I: en las instrucciones: 1-2, 7-8, 10-11
- Dependencia datos tipo II: en la instrucción 6-7
- Dependencia de control: en la instrucción 11

b) Al hacer el conteo de instrucciones a ejecutar:

- Se logra evitar las burbujas por cualquier dependencia datos tipo I gracias al *data forwarding*
- No hay posibilidad de reordenar instrucciones para la dependencia de datos tipo II, por lo que se perderá siempre 1 ciclo de reloj.
- La Previsión de bifurcación dará el siguiente resultado considerando el estado inicial (predict not taked again) y el cumplimiento real de la bifurcación: 9 veces se cumplirá y una no.

Estado	NTA	NT	TA	T	T	T	T	T	T	T
Previsión	0	0	1	1	1	1	1	1	1	1
Real	1	1	1	1	1	1	1	1	1	0
Penalidad	2	2	0	0	0	0	0	0	0	2

Vemos que el predictor se equivocará 3 veces y se requerirá penalidad de 6 ciclos de reloj.

- Se ejecutan 4 instrucciones iniciales, se entrra al bucle (7 instruc) que correrá 10 veces enteras (hasta la última instrucción, por lo que habrán 70 + 4 instrucciones entrando al pipeline (74 ciclos de reloj),
- Analizando las dependencias, se tiene una dependencia de datos tipo II no resuelta que se ejecuta 10 veces (en el bucle) sumando 10 ciclos adicionales, las dependencias datos tipo I están resueltas, y la dependencia de control usando el predictor de 2bits genera 6 ciclos adicionales por sus errores. Así se suman 16 ciclos adicionales por las dependencias
- Se totalizan 74 + 16 = 90 ciclos de reloj
- No se está considerenando los ciclos de reloj de drenaje de las últimas instrucciones

c) El bucle es muy previsible, el mecanismo de un bit o el de dos bits iniciando por estados de cumplimiento sólo tendrán un error. Otra opción es desdoblar el bucle, que se puede saber a priori que se ejecutará 10 veces.

I/O SYSTEMS

Pregunta 4: **[4pt, Habilidades: b2,i3]**

Sean dos HDD (sistema y usuario) que gastan 1100 ciclos de reloj en cada atención por interrupción y disparan 750k interrupciones durante operaciones de transferencia. Sea un procesador de 2GHz, y sea que el disco de sistema tiene un uso de aprox 8% y el disco de usuario de 2%. Calcule el porcentaje de procesamiento que el CPU dedica a atender esas interrupciones

Son dos discos duros con los mismos parámetros de funcionamiento y porcentajes diferentes, luego podemos estimar el uso de CPU de la siguiente forma:

$$\begin{aligned} CPU\% &= \frac{(0.08+0.02)\times 1100\times 750000}{2\times 10^9} \\ &= \frac{0.1\times(825\times 10^6)}{2\times 10^9} \\ &= \frac{825\times 10^5}{2\times 10^9} \\ CPU\% &= 4.125\% \end{aligned}$$

ALL TOPICS

Pregunta 5: **[4pt, Habilidades: b2,i3]**

Responda con verdadero o falso a las siguientes afirmaciones:

a) En *pipelines* avanzados (aquellos con unidades ejecutoras especializadas), el uso de ejecución especulativa es necesario para aumentar el rendimiento. [ ]

b) El uso de memoria caché funciona bien gracias a la característica de localidad global (accesos a una dirección suelen estar agrupados en el tiempo) que ocurre en los códigos ejecutados en una CPU [ ]

c) Cualquier de conjunto de instrucciones (MMX/SSE) corresponde al tipo de computador paralelo (clasificación de Flynn) MISD [ ]

d) El código de Fibonacci de la Pregunta 3 puede acelerarse bastante si usamos la extensión de instrucciones MMX que ejecutan operaciones aritméticas enteras simultaneas en hasta 4 registros de 16 bits u 8 registros de 8 bits.[ ]

a) En *pipelines* avanzados (aquellos con unidades ejecutoras especializadas), el uso de ejecución especulativa es necesario para aumentar el rendimiento. [V]

Es correcta porque en esos pipelines es más facil dejar morir instrucciones sin actualizar resultados que quitarlas y recolocar otras

b) El uso de memoria caché funciona bien gracias a la característica de localidad global (accesos a una dirección suelen estar agrupados en el tiempo) que ocurre en los códigos ejecutados en una CPU [V]

Es correcto porque es una de las características que permiten hacen que los datos en caché sean usados durante un tiempo sin fallar

c) Cualquier de conjunto de instrucciones (MMX/SSE) corresponde al tipo de computador paralelo (clasificación de Flynn) MISD [F] *Falso porque esos tipos de instruciones son las llamadas instrucciones vectoriales, sería la clasificación SIMD (single instruction, multiple data)*

d) El código de Fibonacci de la Preg. 3 puede acelerarse bastante si usamos la extensión de instrucciones MMX que ejecutan operaciones aritméticas enteras simultaneas en hasta 4 registros de 16 bits u 8 registros de 8 bits. [F]

No ocurirrá esta aceleración grande porque los valores en fibonacci dependen de resultados inmediatamente anteriores $f_{t+1} = f_t - f_{t-1}$