Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Arquitectura de Computadoras para Ingeniería

Trabajo Práctico N° 6
Multiple-Issue Processors
Primer Cuatrimestre de 2016

Ejercicios

1. Considerando el siguiente fragmento de código:

$$
\begin{array}{lll}
I_1: & \mathsf{ADD} & \mathsf{R1, R2, R3} \\
I_2: & \mathsf{LD} & \mathsf{R4, (R5)} \\
I_3: & \mathsf{SUB} & \mathsf{R7, R1, R9} \\
I_4: & \mathsf{MUL} & \mathsf{R5, R4, R4} \\
I_5: & \mathsf{SUB} & \mathsf{R1, R12, R10} \\
I_6: & \mathsf{ST} & \mathsf{(R13), R14} \\
I_7: & \mathsf{OR} & \mathsf{R15, R14, R12}
\end{array}
$$

   Identifique las **dependencias involucradas** y muestre las instrucciones que serán despachadas por ciclo considerando que se pueden procesar:

   a) Dos instrucciones de forma simultánea
   b) Cuatro instrucciones de forma simultánea

   Asumir que el procesador posee las suficientes unidades de ejecución de forma de evitar conflictos estructurales y que cada instrucción posee una latencia de un ciclo. A su vez, considerar que no existen restricciones sobre el tamaño de la ventana de instrucciones.

   Obs: no utilice diagramas de *Gantt*, simplemente muestre qué instrucciones son despachadas en cada ciclo.

2. En un procesador *Superescalar (Dynamic Issue)* con **ejecución en orden** se realiza el fetch de un conjunto w de instrucciones en un mismo ciclo y dinámicamente se las despacha *(issue)* acorde a que estas sean independientes entre sí y no existan conflictos a nivel de recursos. En este contexto, muestre la ejecución de la siguiente secuencia de instrucciones utilizando un diagrama de *Gantt*:

$$
\begin{array}{lll}
I_1: & \mathsf{LD} & \mathsf{R1, (R2)} \\
I_2: & \mathsf{ADD} & \mathsf{R3, R1, R4} \\
I_3: & \mathsf{SUB} & \mathsf{R5, R6, R7} \\
I_4: & \mathsf{MUL} & \mathsf{R8, R9, R10}
\end{array}
$$

   Asumir que la operaciones de acceso a memoria poseen una latencia de dos ciclos, mientras que las operaciones aritméticas poseen una latencia de un ciclo. Considerar que el procesador puede despachar hasta dos instrucciones por ciclo y posee tres unidades funcionales: una de suma, una de producto y una para el cálculo de direcciones efectivas.

3. Dada la siguiente secuencia de instrucciones:

$$
\begin{array}{rlll}
I_1: & \mathsf{LD} & \mathsf{R1, (R2)} \\
I_2: & \mathsf{SUB} & \mathsf{R4, R5, R6} \\
I_3: & \mathsf{ADD} & \mathsf{R3, R1, R7} \\
I_4: & \mathsf{MUL} & \mathsf{R8, R3, R3} \\
I_5: & \mathsf{ST} & \mathsf{(R11), R4} \\
I_6: & \mathsf{ST} & \mathsf{(R12), R8} \\
I_7: & \mathsf{ADD} & \mathsf{R15, R14, R13} \\
I_8: & \mathsf{SUB} & \mathsf{R10, R15, R10} \\
I_9: & \mathsf{ST} & \mathsf{(R9), R10}
\end{array}
$$

   Resolver los siguientes incisos asumiendo que la arquitectura cuenta con un procesador superescalar con **ejecución en orden**, el cual cuenta con dos unidades de ejecución, es decir que se pueden despachar dos instrucciones (de cualquier tipo) por ciclo (w=2). Asumir que las instrucciones de acceso a memoria poseen una latencia de dos ciclos, mientras que las restantes poseen una latencia de un ciclo.

   a) Esquematizar en un diagrama de Gantt el solapamiento entre instrucciones, indicando claramente en qué ciclo de reloj se despacha cada instrucción y por qué.

   b) Indique el tiempo de despacho y el tiempo de ejecución de la secuencia de instrucciones.

4. Repetir el ejercicio 3, esta vez considerando que la arquitectura posee dos unidades funcionales para el cálculo de direcciones efectivas de 1 ciclo, una unidad funcional de suma de 1 ciclo y una unidad multiplicadora de 3 ciclos. A su vez, suponer que la arquitectura posee *forwarding* y hacer explícitos los *bypasses* necesarios.

5. Sea un procesador Superescalar con ejecución en orden, el cual consta de 4 unidades de ejecución ($w = 4$) capaces de ejecutar cualquier operación. Dada la siguiente secuencia de instrucciones:

$$
\begin{array}{rlll}
I_1: & \mathsf{ADD} & \mathsf{R1, R2, R3} \\
I_2: & \mathsf{SUB} & \mathsf{R5, R4, R5} \\
I_3: & \mathsf{LD} & \mathsf{R4, (R7)} \\
I_4: & \mathsf{MUL} & \mathsf{R4, R4, R4} \\
I_5: & \mathsf{ST} & \mathsf{(R1), R4} \\
I_6: & \mathsf{LD} & \mathsf{R9, (R10)} \\
I_7: & \mathsf{LD} & \mathsf{R11, (R12)} \\
I_8: & \mathsf{ADD} & \mathsf{R11, R11, R12} \\
I_9: & \mathsf{MUL} & \mathsf{R11, R11, R11} \\
I_{10}: & \mathsf{ST} & \mathsf{(R12), R11}
\end{array}
$$

   Esquematizar en un diagrama de Gantt el solapamiento entre instrucciones, indicando claramente en qué ciclo de reloj se despacha cada instrucción y por qué. Indique el tiempo de despacho y el tiempo de ejecución de la secuencia de instrucciones.

   Obs.: Asumir que las operaciones de suma y resta poseen una latencia de 1 ciclo, las operaciones de producto 3 ciclos y las de acceso a memoria 2 ciclos.

6. En un procesador *Superescalar (Dinamic Issue)* con ejecución **fuera de orden**, se realiza el fetch de un conjunto w de instrucciones y se las despacha (issue) en un determinado ciclo. Los conflictos a nivel de recursos de datos no frenarán el pipeline, solo a las intrucciones con dependencias.

   Rehacer los ejercicios 4 y 5 para las mismas condiciones pero con ejecución fuera de orden.

7. **VLIW: Very Long Instruction Word** Para explotar el paralelismo a nivel de instrucciones (ILP), los procesadores VLIW utilizan el *compilador* para determinar qué instrucciones pueden ejecutarse simultáneamente. En estos procesadores, cada instrucción especifica un conjunto de operaciones independientes entre sí que pueden ejecutarse en paralelo. Cada operación se corresponde a una instrucción en un procesador superescalar o secuencial. La cantidad de operaciones por instrucción en un VLIW es igual a la cantidad de unidades de ejecución del procesador (w). El hardware no realizará ningún tipo de chequeo de dependencias. A diferencia de los procesadores superescalares, el ***compilador*** analizará **todas** las posibilidades que brinda el código completo.

   Mostrar cómo el compilador planificaría el código del ejercicio 5 considerando un procesador VLIW, el cual define cuatro *slots* por instrucción y está conformado por dos unidades funcionales para el cálculo de operaciones enteras de un ciclo y dos unidades para acceso a memoria de dos ciclos.

   Obs.: El compilador puede planificar instrucciones con dependencia WAR fuera de orden, siempre y cuando la instrucción que lea el registro se despache antes que la que lo escribe termine, porque el registro no cambia efectivamente hasta que la escritura finalice.

8. Un procesador VLIW define cinco *slots* por instrucción y está conformado por dos unidades funcionales para el cálculo de operaciones enteras de dos ciclos, una unidad funcional para el cálculo de operaciones de punto flotante de dos ciclos y dos unidades para acceso a memoria de tres ciclos. En este contexto, mostrar el formato de la instrucción y determinar cómo el compilador planificaría el siguiente código:

I_1 :	ADDI	R1, R2, R3	I_{10} :	LD	F21, (R23)
I_2 :	MULI	R4, R5, R1	I_{11} :	LD	R24, (R25)
I_3 :	SUBD	F1, F3, F4	I_{12} :	ADDD	F17, F18, F19
I_4 :	LD	R2, (R21)	I_{13} :	SUBD	F15, F21, F23
I_5 :	SUBI	R8, R4, #8	I_{14} :	ADDI	R12, R21, R3
I_6 :	LD	F11, (R4)	I_{15} :	MULI	R21, R24, R28
I_7 :	ADDI	R9, R15, R16	I_{16} :	MULD	F23, F24, F28
I_8 :	SUBI	R15, R16, R17	I_{17} :	SUBD	F18, F4, F21
I_9 :	LD	R21, (R22)			