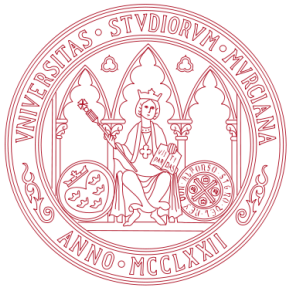

Universidad de Murcia

Facultad de Informática

TÍTULO DE GRADO EN
INGENIERÍA INFORMÁTICA

Ampliación de Estructura de Computadores

Tema 2: Segmentación Básica

Boletín de ejercicios

CURSO 2024 / 2025

Departamento de Ingeniería y Tecnología de Computadores

Área de Arquitectura y Tecnología de Computadores

Ejercicios Básicos

1. Para el procesdor DLX sin segmentar estudiado en clase, se ha observado que el tiempo mínimo necesario para completar cada una de las cinco etapas del cauce es de 10 nseg, 8 nseg, 10 nseg, 10 nseg y 7 nseg, respectivamente. Asumiendo que el 30 % de las instrucciones ejecutadas son de tipo ALU (tardan 5 ciclos), el 20 % son saltos (tardan 4 ciclos), el 15 % son instrucciones de almacenamiento en memoria (tardan 4 ciclos), el 35 % son instrucciones de carga (tardan 5 ciclos), y dado que la sobrecarga introducida por la segmentación haría que el tiempo de ciclo aumentase en 1 nseg con respecto a la versión multiciclo, se pide:

   a) ¿Cuál sería la frecuencia de funcionamiento del procesador segmentado?

   b) ¿Qué aceleración obtendría un procesador segmentado ideal (sin riesgos) con respecto a la versión multiciclo sin segmentar?

   c) ¿Qué aceleración obtendría un procesador segmentado ideal (sin riesgos) con respecto a la versión monociclo?

2. ¿Cuál es el tiempo de ejecución (en ciclos) de la siguiente secuencia de instrucciones en el cauce de cinco etapas visto en clase?

```
ADD R3,R4,R5
SUB R7,R3,R9
MUL R8,R9,R10
ADD R4,R8,R12
```

   ¿Puede el compilador mejorar el tiempo de ejecución de la secuencia de instrucciones planificando las instrucciones sin cambiar el resultado del cálculo? Si es así, muestra la secuencia de instrucciones con el menor tiempo de ejecución e indica cuál es ese tiempo.

3. Suponiendo que el valor inicial del R4 es R2+792, y usando el siguiente código:

```
Loop: LW    R0,0(R2)
      LW    R1,0(R3)
      MULT  R0,R0,R1
      ADD   R6,R0,R6
      ADDI  R2,R2,#8
      ADDI  R3,R3,#8
      SUB   R5,R4,R2
      BNEZ  R5,Loop
```

   Muestra la temporización de esta secuencia de instrucciones para el cauce DLX estudiado, donde los riesgos se manejan deteniendo el cauce y la condición y destino del salto no se resuelven hasta el final de la etapa MEM. Si todas las referencias a memoria aciertan en cache, ¿cuántos ciclos tardará en ejecutarse este bucle?.

4. Tenemos un procesador con el siguiente cauce: *F1 F2 D1 D2 X1 X2 M1 M2 WB*, que ejecuta instrucciones del repertorio del DLX y donde se determina el destino de los saltos en la etapa *D1*. La distribución de instrucciones es la siguiente: 40 % ALU, 20 % cargas, 10 % almacenamientos y 30 % saltos condicionales. El repertorio de instrucciones incluye dos tipos de instrucciones de salto condicional, saltos en los que el operando a comparar con cero está en un registro (`BNEZ Rx, Label`), que representan el 80 % de las instrucciones de salto y que son resueltas en la etapa

X1, y saltos para los que el operando está en memoria (`BNEZ 0(Rx), Label`), que constituyen el 20 % restante de las instrucciones de salto y que se resuelven en *M2*. Considerando que se utiliza la detención del cauce para resolver los riesgos de control y teniendo en cuenta únicamente los ciclos de detención ocasionados por las instrucciones de salto, calcular el CPI total de la máquina descrita.

Ejercicios Adicionales

5. Tenemos un procesador DLX (llamémosle A), con pipeline *F D X M W* como el visto en clase, con cálculo del destino y la condición en la etapa *D*. En el código que ejecutamos encontramos la siguiente distribución de instrucciones: 30 % ALU (1.1 ciclos, incluida penalización por riesgos de datos), 30 % cargas (1.2 ciclos, incluida penalización por riesgos de datos), 20 % almacenamientos (1.2 ciclos, incluida penalización por riesgos de datos) y 20 % saltos condicionales (1 ciclo sin incluir penalización por riesgos de control). Al analizar el código identificamos que el siguiente fragmento aparece muy frecuentemente, formando el 30 % de las instrucciones ejecutadas:

```
LW    R1,0(R2)
LW    R3,0(R4)
SLT   R5,R1,R3
SW    0(R2),R3
SW    0(R4),R1
BNEZ  R5,Label
```

   Así que decidimos que nos interesa rediseñar nuestro procesador y obtener el DLX B, con pipeline $F\ D\ X_1\ M_1\ M_2\ X_2\ M_3\ M_4\ W$ que ejecuta el mismo código de antes y en las mismas condiciones, pero que nos permite incluir una nueva instrucción de salto llamada `XBLT 0(R2),0(R4),Label`, que hace lo mismo que el código anterior pero que consume 1.5 ciclos antes de incluir la penalización por salto. La condición de este salto se evalúa en la etapa X_2. El procesador puede seguir ejecutando la instrucción de salto antigua en las mismas condiciones porque conserva la comprobación original también. Como efecto secundario, las cargas pasan a tardar 1.4 ciclos y el nuevo ciclo es el 70 % del antiguo. El resto de operaciones mantienen el mismo CPI.

   Compara los rendimientos de las dos máquinas cuando incluimos la penalización de los saltos.

6. Una máquina se dice que está subsegmentada si se puede subdividir el cauce en más etapas sin variar apreciablemente el número de detenciones. Vamos a suponer que el cauce del DLX entero se ha cambiado a cuatro etapas al juntar las etapas EX y MEM, lo que también va a provocar que tengamos que aumentar el ciclo de reloj en un 50 %. En ambos casos el salto se resuelve en la etapa ID. ¿Cuánto más rápido será el DLX convencional frente al DLX subsegmentado para código entero? Suponer que en el cauce original el 9 % de las instrucciones provocan detenciones: 5 % debido a cargas seguidas inmediatamente por una operación que necesita el valor cargado y 4 % debido a saltos.

7. Tenemos una máquina con un conjunto de instrucciones que incluye tres tipos de operaciones de ALU dependiendo de la naturaleza de sus operandos: ALU_{RRR}, ALU_{RRM} y ALU_{MRM}. Donde *RRR* indica que los tres operandos son registros, y una *M* indica que ese operando se encuentra en memoria. La distribución de instrucciones es la siguiente: 50 % ALU, 20 % cargas (1.2 ciclos), 10 % almacenamientos (1.3 ciclos) y 20 % saltos (1.2 ciclos). Dentro de las operaciones ALU encontramos 50 % de tipo *RRR* (1.1 ciclos), un 30 % de tipo *RRM* (1.2 ciclos) y un 20 % de tipo *MRM* (1.3 ciclos). Sabiendo esto, nos ofrecen las siguientes alternativas:

   a) Sustituir las instrucciones ALU_{RRM} por una carga y una de tipo ALU_{RRR}, y las instrucciones ALU_{MRM} por una carga, una instrucción ALU_{RRR} y un almacenamiento. De esta forma podemos hacer la frecuencia de reloj un 50 % mayor.

b) Utilizar un compilador con un optimizador más avanzado que reutilizara mejor los registros y sustituyera el 20 % de las instrucciones ALU_{RRM} y el 30 % de las ALU_{MRM} por instrucciones ALU_{RRR}.

Con los datos anteriores, se pide determinar qué configuración ofrecería más rendimiento y en qué factor.

8. Tenemos un procesador con la estructura de pipeline que se muestra a continuación:

   $F1\ F2\ ID\ EX\ MEM\ WB$

   Todas las etapas se encuentran segmentadas, salvo la de ejecución (EX) que tiene una duración de 3 ciclos. Hay un 75 % de saltos que se resuelven en la tercera etapa (ID), mientras que los restantes se resuelven en la etapa (MEM). Se ha comprobado que por término medio los programas ejecutados tienen la siguiente distribución de instrucciones: 20 % de ALU, 20 % de cargas, 10 % de almacenamientos y 50 % de saltos condicionales, de los cuales el 60 % son saltos tomados. Los saltos se manejan deteniendo el cauce. Se ha visto, además, que:

   - La cuarta parte de las cargas van seguidas inmediatamente por una instrucción (ej. ALU) que necesita el valor de la carga.
   - La mitad de las instrucciones ALU van seguidas inmediatamente por otra instrucción que necesita el dato y la otra mitad van seguidas inmediatamente por una instrucción de salto condicional que requiere el dato calculado por ellas.

   Considerando únicamente las dependencias de datos descritas en los ítems de arriba, describe los riesgos que podrán aparecer en el cauce poniendo un ejemplo de instrucción/secuencia de instrucciones, y cuantifica la incidencia que cada uno de estos riesgos identificados tendría sobre el rendimiento final (cuánto aumenta el CPI).