Trabajo Práctico Número 5

Arquitectura de Computadoras 6/05/2026

Instrucciones

- Los ejercicios propuestos en el presente trabajo práctico pueden ser resueltos en forma individual o grupal.
- El planteo de la solución debe realizarse basándose en lo aprendido en las clases teóricas.
- Puede utilizar las clases de consulta para consultar ejercicios de enunciado y verificar la validez de la solución obtenida.
- Posteriormente, se tomará una evaluación con ejercicios similares a los de este práctico, la cual deberá ser resuelta en forma individual. La fecha de evaluación se encuentra en el sitio web de la materia.

Conceptos involucrados

- Riesgos de control: predicción de saltos estática y dinámica.
- Caminos de datos superescalares.
- Emisión múltiple de instrucciones.
- Ejecución de instrucciones fuera de orden.
- Paralelismo a nivel de instrucciones (ILP).
- Manejo de interrupciones y excepciones.
- Optimización de código mediante desenrollado de lazos.
- Renombramiento de registros.

Problemas propuestos

1) Indique cuáles de las siguientes técnicas caracteriza al paralelismo a nivel de instrucción (ILP).
   a) Ejecución en orden.
   b) Varias unidades en paralelo en un procesador.
   c) Ejecución fuera de orden.
   d) Varios procesadores en paralelo.
   e) Ejecución especulativa.
   f) Predicción de saltos.
   g) Renombramiento de registros.
   h) Uniciclo.
   i) Software pipelining.
   j) Desenrollado de lazos.
   k) VLIW.

2) Dados los dos segmentos de código que se muestran a continuación, indique si se benefician más con una ejecución superescalar, o si se beneficiarían más con una ejecución fuera de orden con renombramiento de registros, o si no se beneficiarían con ninguno de los dos casos. Justifique lo más detalladamente su respuesta.

a)
```
add s0, s1, t0
add s2, s3, t0
add s4, s5, t0
add s6, s7, t0
```

b)
```
lw  s1, 0(s2)
add s2, s1, s1
lw  s1, 0(s3)
lw  s3, 0(s4)
add s3, s1, s3
add s1, s2, s3
```

3) Considere la siguiente pieza de código en C:

```
for (i = 0; i < 1000000; i++) {
    a = random(100);
    if (a >= 50) {
        ...
    }
}
```

Suponga que la función Random(N) retorna un número aleatorio uniformemente distribuido entre 0 y N-1 inclusive. Considere la instrucción de salto asociada a la sentencia **if**. Indique cuál o cuáles de las siguientes opciones elegiría y justifique su respuesta.

a) No usar predicción y esperar hasta que se defina si se salta o no.
b) Predecir siempre el salto como tomado y rehacer en caso de equivocación.
c) Predecir siempre el salto como no tomado y rehacer en caso de equivocación.
d) Usar tabla de saltos para predecir, indique qué algoritmo tomaría para saltos en este caso

4) *[Recomendado]* Se tiene un procesador que implementa el ISA RV32I del RISC-V con un diseño en pipeline de cinco etapas. Sobre este procesador se ejecuta el siguiente lazo:

```
    .data
vector:
    .word   72,57,39,48,46,88,2,91,0

    .text
    .globl  main

    la a0, vector       # Carga la direccion base del vector
    li a1, 10           # Carga la cantidad de elementos del vector
    li a2, 0            # Inicializa el contador de elementos impares
lazo:
    addi a1, a1, -1     # Decrementa el contador de elementos
    beqz a1, fin        # Termina ciclo si elemento es cero
    lw   t0, 0(a0)      # Carga el elemento a utilizar
    addi a0, a0, 4      # Incrementa puntero a siguiente elemento
    andi t0, t0, 1      # Enmascara ultimo bit para determinar paridad
    beqz t0, par        # Salta si el numero es par
    addi a2, a2, 1      # Incrementa contador de impares
par:
    j lazo              # Repite lazo para siguiente elemento
fin:
```

a) Si el procesador sigue el diseño del primer procesador segmentado visto en clase, pero incorpora todos los adelantamientos necesarios para resolver los riegos de datos, ¿cuántos ciclos demora en este procesador la ejecución del lazo propuesto?
b) Si el procesador sigue el diseño del mejor procesador segmentado visto en clase, ¿cuántos ciclos demora ahora la ejecución del lazo propuesto?
c) Suponga ahora que se adiciona una unidad de predicción de saltos que posee un 100 % de acierto, de manera tal que cuando un salto está en la etapa de Decode entonces la instrucción que ingresa al Fetch es siempre la correcta. En este caso, ¿cuántos ciclos demora en este procesador la ejecución del lazo propuesto?
d) Suponga ahora que se utiliza una predicción estática de saltos, en la cual los saltos hacia atrás se predicen como tomados y los saltos hacia adelante se predicen como que no serán tomados. En este caso, ¿cuántos ciclos demora en este procesador la ejecución del lazo propuesto?
e) Suponga ahora que se incluye una tabla para la predicción dinámica de los saltos con dos entradas y un solo bit para mantener la historia de cada salto. Como predicción inicial se utiliza el mismo criterio del apartado anterior. ¿cuántos ciclos demora en este procesador la ejecución del lazo propuesto?¿Cuál es la tasa de aciertos en la predicción?
f) Si la tabla de predicción del ejercicio anterior se amplia a cuatro entradas ¿Cuál es la nueva tasa de aciertos para la predicción?¿Cuánto es la mejora respecto al caso anterior?

5) *[Profundización]* La frecuencia con la que se ejecutan los saltos condicionales, junto con la precisión del predictor de saltos, determinan la pérdida de rendimiento en el pipeline debido a los saltos mal pronosticados. Suponiendo que el desglose de las instrucciones en categorías es el siguiente:

Tipo R	BEQZ/BNEZ	JAL	LW	SW
40 %	25 %	5 %	25 %	5 %

Además, suponiendo las siguientes precisiones para los diferentes predictores de saltos:

Siempre tomado	No tomado nunca	2-Bit
45 %	55 %	85 %

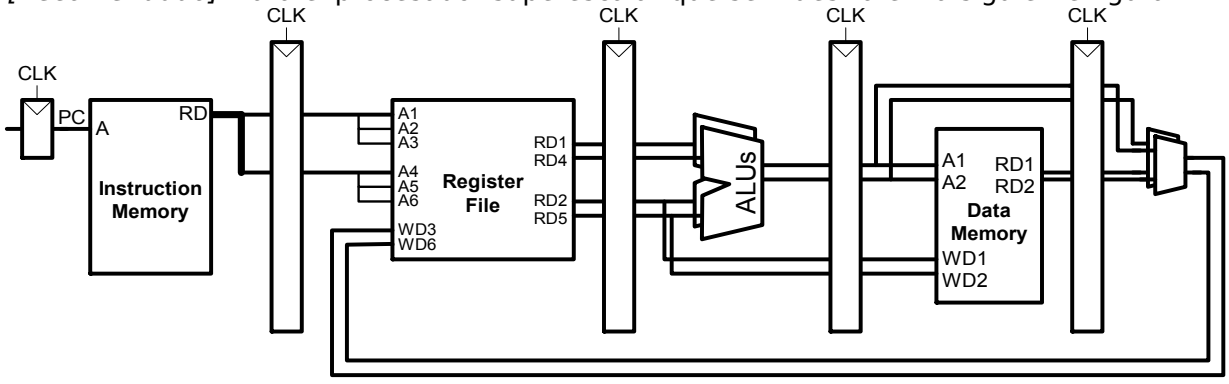
a) Suponiendo que se utiliza el mejor procesador segmentado visto en clase, en el cual los saltos se resuelven en la etapa ID ¿cuál es el CPI adicional debido a los saltos mal predichos cuando se utiliza la predicción “Siempre tomado”?
b) Repita el ejercicio del apartado anterior para la predicción “No tomado nunca”.
c) Repita el ejercicio del apartado anterior para el predictor de 2 bits.
d) Con el predictor de 2 bits, ¿qué aceleración se lograría con respecto al apartado anterior si pudiéramos convertir la mitad de las instrucciones de salto en una instrucción ALU?. Suponga que las instrucciones predichas correcta e incorrectamente tienen la misma probabilidad de ser reemplazadas.
e) Con el predictor de 2 bits, ¿qué aceleración (nuevamente, con respecto al apartado c) se lograría si pudiéramos convertir la mitad de las instrucciones de salto de manera que se remplace una instrucción de salto con dos instrucciones ALU? Suponga que las instrucciones predichas correcta e incorrectamente tienen la misma probabilidad de ser reemplazadas.
f) Algunas instrucciones de salto son mucho más predecibles que otras. Si se sabe que el 80 % de todas las instrucciones de salto ejecutadas son saltos hacia atrás pertenecientes a lazos, y que por lo tanto se los puede predecir siempre correctamente, ¿cuál sería la precisión del predictor de 2 bits para el 20 % restante de las instrucciones de salto?

6) *[Recomendado]* A fin de incrementar la performance del procesador en pipeline visto en clase, se va a diseñar un nuevo procesador consistente en dos pipes en paralelo. Este procesador será capaz de procesar dos instrucciones RV32-I en paralelo.
   a) ¿Qué riesgos estructurales podría haber entre dos instrucciones en dos pipelines en paralelo y cómo los resolvería? Considere Memoria de Instrucciones, Memoria de Datos, Banco de Registros, ALU, etc.
   b) ¿Es necesario que los circuitos de adelantamiento pasen información de un pipe al otro? No se pide diseñarlos. Justifique su respuesta.
   c) ¿Entre dos instrucciones que se ejecuten en paralelo podrían existir riesgos WAR? En caso positivo dé un ejemplo, caso contrario explique su respuesta.
   d) ¿Entre dos instrucciones que se ejecuten en paralelo podrían existir riesgos WAW? En caso positivo dé un ejemplo, caso contrario explique su respuesta.
   e) A fin de analizar los riesgos RAW, suponga que cada pipe contiene todos los circuitos de adelantamiento necesarios, ¿Existen riesgos RAW?. En caso positivo dé un ejemplo, caso contrario explique su respuesta.
   f) Analice los dos siguientes segmentos de programa en que las dos instrucciones se podrían paralelizar e indique si el resultado sería correcto o no:

```
add  t0, t1, t2          bnez t0, salto
bnez t0, salto           add  t0, t1, t2
```

   g) ¿Se debe habilitar a los dos pipes para ejecutar instrucciones de salto, o solo a uno? Explique.
   h) En caso de salto, ¿cuál es la penalidad y cuántas burbujas se crearían como máximo?

7) *[Recomendado]* Para el procesador superescalar que se muestra en la siguiente figura:



Y sabiendo que ambos caminos son para números enteros, se pide calcular el IPC para los siguientes segmentos de código:

a)
```
lw  t0, 40(s0)
add t1, s1, s2
sub t2, s1, s3
and t3, s3, s4
or  t4, s1, s5
sw  a5, 80(s0)
```

b)
```
lw  t0, 40(s0)
add t1, t0, s1
sub t0, s2, s3
and t2, s4, t0
or  t3, t2, s6
sw  s7, 80(t3)
```

Considere que todos los caminos de adelantamiento están resueltos. En caso de que encuentre riesgos en los segmentos de código de (a) y (b), deberá clasificar los mismos (RAW, WAW, WAR) y reescribir el código empleando renombramiento de registros y reordenamiento de instrucciones y recalcular IPC.

8) Considere el siguiente segmento de programa en lenguaje ensamblador de RISC-V:

```
(1)  add  x1, x2, x3
(2)  sub  x5, x4, x5
(3)  lw   x4, 0(x13)
(4)  mul  x4, x4, x4
(5)  sw   x4, 0(x13)
(6)  lw   x9, 0(x10)
(7)  lw  x11, 0(x12)
(8)  add x11, x11, x12
(9)  mul x11, x11, x11
(10) sw  x11, 0(x12)
```

a) Realice el diagrama de precedencia para el segmento de programa mostrado.
b) Si se ejecuta el código en el procesador segmentado visto en clase, ¿cuántos ciclos tarda en ejecutarse?
c) Si se ejecuta el código en un procesador superescalar que puede ejecutar hasta cuatro instrucciones en paralelo, pero que finaliza las instrucciones *en el mismo orden* en que llegan, ¿cuántos ciclos tarda en ejecutarse?
d) Si se ejecuta el código en un procesador superescalar que puede ejecutar hasta cuatro instrucciones en paralelo, y que puede finalizar las instrucciones *en cualquier orden*, ¿cuál es el mínimo tiempo que se necesita para ejecutar el código?

9) *[Recomendado]* Dado el siguiente código, en el cual todas las instrucciones duran 1T y no requiere paradas después de un instrucción de carga:

```
lw  r1, 0(r2)
add r3,r4,r1
sub r4,r5,r6
mul r7,r4,r8
sll r8,r9,r10
sub r11,r8,r12
and r12,r13,r14
sw  r12,0(r15)
```

a) Indique cuáles son las dependencias WAR y WAW, y en qué registro(s), suponiendo que todas las instrucciones podrían ejecutarse al mismo tiempo.
b) Reescriba el código renombrando los registros para evitar estas dependencias.
c) ¿En cuántos T se podría ejecutar este código fuera de orden y con todo el paralelismo que fuera necesario? ¿Cuánto paralelismo sería necesario?

10) *[Recomendado]* En el trabajo de Laboratorio 1, se vio un ejercicio en el cual se buscaba una implementación del algoritmo GEMM para multiplicar matrices. A continuación, se muestra una versión optimizada del lazo más interno de dicho algortimo:

```
L3:   lw   s11, 0(t2)     # s11 = B[k][j]
      lw   s10, 0(t1)     # s10 = A[i][k]
      mul  s11, s10, s11  # s11 = A[i][k] * B[k][j]
      add  s0, s0, s11    # s0 = C[i][j] + A[i][k] * B[k][j]

      add  t2, t2, s3     # t2 += N (siguiente elemento de B)
      addi t1, t1, 4      # t1++ (siguiente elemento de A)
      bltu t2, s5, L3     # si k < N, saltamos a L3
```

Suponga que el lazo anterior se ejecuta un gran número de veces, y por lo tanto se desea optimizarlo utilizando la técnica de desenrollado de lazos.

a) ¿Cuántos ciclos demora cada iteración del lazo y cuántos bytes ocupa en memoria?
b) Desenrollar 1 sola vez y reordenar el código. ¿Cuántos ciclos demora ahora cada iteración y cuántos bytes ocupa en memoria?
c) Desenrollar 3 veces y reordenar el código. ¿Cuántos ciclos demora ahora cada iteración y cuántos bytes ocupa en memoria?

11) *[Recomendado]* Si en un procesador superescalar de emisión doble con excepciones precisas ocurre una excepción en ambas instrucciones en la misma etapa ¿Como se determina cual de las excepciones es la que se registra como causa?

12) *[Profundización]* Este ejercicio explora cómo el manejo de excepciones afecta el diseño de un procesador segmentado. Los tres primeros problemas de este ejercicio se refieren a las dos instrucciones siguientes:

Instrucción 1	Instrucción 2
`beqz x11, ETIQUETA`	`lw x11, 0(x12)`

a) ¿Qué excepciones puede desencadenar cada una de estas instrucciones? Para cada una de estas excepciones, especifique en que etapa del pipeline en la que se detecta.
b) Si hay una dirección de la rutina de servicio para cada excepción, muestre cómo se debe cambiar la organización del pipeline para poder manejar esta excepción. Puede suponer que las direcciones de estos controladores se conocen cuando se diseña el procesador.
c) Si la segunda instrucción se obtiene inmediatamente después de la primera instrucción, describa lo que sucede en el pipeline cuando la primera instrucción provoca la primera excepción que enumeró en el apartado a. Muestre el diagrama de ejecución del pipeline desde el momento en que se obtiene la primera instrucción hasta el momento en que se completa la primera instrucción de la rutina de servicio de la excepción.
d) En el manejo de excepciones vectorizado, la tabla de direcciones de las rutinas de servicio de las excepciones está en la memoria de datos a partir de una dirección conocida (fija). Cambie la implementación del procesador para incluir este mecanismo de manejo de excepciones. Repita el apartado c utilizando esta versión modificada y el manejo de excepciones vectorizado.
e) ¿Qué cambios debería realizar en la implementación anterior si ahora desea que programador pueda ubicar el vector de interrupciones en diferentes zonas de memoria?

Ejercicio a Desarrollar en Clase Práctica (Problema del primer parcial 2019)

13) Se desea comparar la performance de distintas variantes de un código que suma dos vectores de memoria A y B, y almacena el resultado en un tercer vector C. Los vectores son de enteros de 32 bits, tienen 100 elementos cada uno, y se ubican en direcciones contiguas de memoria, comenzando por la dirección 0. El código en lenguaje ensamblador del RISC-V que realiza la tarea anteriormente descrita es el siguiente:

```
      li   a0,0        # Puntero al primer elemento
      li   a1,400      # Limite de un vector
loop: lw   t0,0(a0)    # Carga del primer vector
      lw   t1,400(a0)  # Carga del segundo vector
      add  t0,t0,t1    # Suma los dos elementos
      sw   t0,800(a0)  # Almacena en el resultado
      addi a0,a0,4     # Incrementa el puntero
      bne  a1,a0,loop  # Compara con el limite
```

a) ¿Cuánto demora en ejecutarse el código en la mejor implementación del procesador segmentado de cinco etapas visto en clase? ¿Cuántas instrucciones promedio por ciclo (IPC) se ejecutan? *Sugerencia: desprecie los tiempos de llenado y vaciado del pipeline.*
b) Desenrolle el código una vez (de manera de que en una iteración 'nueva' se realicen dos iteraciones 'viejas'), indique cuánto es la duración en este caso, cuánto es el IPC, y cuánto es la aceleración con respecto al tiempo calculado en el apartado (a).
c) Desenrolle el código tres veces (de manera de que en una iteración 'nueva' se realicen cuatro iteraciones 'viejas'), indique cuánto es la duración en este caso, cuánto es el IPC, y cuánto es la aceleración con respecto al tiempo calculado en el apartado (a).
d) Repita el apartado (a), pero considerando que el código se ejecuta en procesador con emisión de instrucciones estática doble, que es capaz de ejecutar simultáneamente una instrucción de tipo Ld/St junto con otra de tipo R/Branch.
e) Repita el apartado (c), pero considerando el procesador dual-issue descrito en el apartado anterior.
f) Uno de sus compañeros sugiere que en un procesador con emisión de instrucciones estática triple, que es capaz de ejecutar simultáneamente una instrucción de tipo Ld/St junto con dos instrucciones de tipo R/Branch se podría mejorar aún más la performance de este código. ¿Qué piensa Ud. al respecto? Justifique su respuesta.

Soluciones propuestas

1) ...

2) El código de la izquierda se beneficia más con una ejecución superescalar, y el de la derecha con ejecución fuera de orden y renombramiento de registros.

3) Opción b o c, ya que son exactamente iguales

4)
   a) Tiempo de ejecución igual a 140 T.
   b) Tiempo de ejecución igual a 90 T.
   c) Tiempo de ejecución igual a 65 T.
   d) Tiempo de ejecución igual a 71 T.
   e) Tiempo de ejecución igual a 71 T.
   f) Tasa de acierto igual 80 % y mejora igual a 1,4 %.

5)
   a) El CPI aumenta un 13.75 %, pasando a 1,1375.
   b) El CPI aumenta un 11.25 %, pasando a 1,1125.
   c) El CPI aumenta un 3.75 %, pasando a 1,0375.
   d) A = 1.018
   e) A = 0.907
   f) Precisión resto de los saltos = 25 %.

6)
   a) La memoria de instrucciones, la memoria de datos y el banco de registros deben ser únicos, pero es necesario ampliar sus puertos de lectura y escritura. Se deben tener 2 ALUs.
   b) Sí, es necesario.
   c) No, no es posible que ocurran riesgos WAR.
   d) Sí podrían ocurrir riesgos WAW.
   e) Sigue existiendo el riesgo RAW del load retardado. Además, se suman nuevos riesgos entre instrucciones que se ejecutan simultáneamente en ambos pipelines.
   f) El primer par de instrucciones no puede paralelizarse, mientras que el segundo par sí puede paralelizarse, y el resultado sería correcto.
   g) Debe habilitarse solo a un pipe para ejecutar saltos.
   h) En el peor caso, habría tres burbujas.

7)
   a) No hay riesgos. El IPC es 2.
   b) Hay 4 riesgos RAW, 1 riesgo WAW y 1 riesgo WAR, que producen que el IPC sea de 1. Renombrando un registro y moviendo 2 instrucciones de lugar, se puede obtener un IPC igual a 1.5.

8)
   a) ...
   b) 12 ciclos T.
   c) 8 ciclos T.
   d) 5 ciclos T.

9)
   a) Hay tres dependencias WAR, y ninguna WAW.
   b) ...
   c) Con 4 unidades en paralelo, se puede ejecutar el código en 2 T.

10)
   a) El programa original demora 8 T por iteración. El código ocupa 28 bytes en memoria.
   b) Desenrollamos el lazo 1 vez a fin de evitar la parada en los **`mul`** dependientes de los **`lw`**. Obtenemos una iteración cada 6T y el programa en este caso ocupa 48 bytes.
   c) Obtenemos una iteración cada 5 T y el programa ocupa 80 bytes.

11) ...

12)

a) La primera instrucción puede generar un dirección de destino inválida (en la etapa ID), mientras que la segunda instrucción puede generar un acceso a una dirección de memoria para la cual no se tiene permisos de lectura (en la etapa MEM).
b) Se debe agrandar el multiplexor que selecciona la fuente de PC, para permitir las nuevas direcciones constantes.
c) Al detectarse la excepción en la etapa ID, se convierte en NOP la segunda instrucción (que estaba en IF), y en el ciclo siguiente se comienza en IF la búsqueda de la primera instrucción del manejador de excepciones.
d) Es necesario incluir una instrucción que permita mover el contenido del registro de causa al banco de registros. Luego, la dirección del manejador de excepciones se puede obtener con el modo de direccionamiento indexado estándar.
e) La dirección base del vector de interrupciones no debe ser fija, sino que deberá estar contenida en un registro especial.