PROGRAMADOR UNIVERSITARIO
LICENCIATURA EN INFORMÁTICA
INGENIERÍA EN INFORMÁTICA
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 8

PRADO, MATIAS SANTIAGO

1. ¿Qué motivó el surgimiento de la arquitectura RISC? Describa 3 diferencias claras entre las arquitecturas CISC y RISC.

El surgimiento de RISC fue motivado por un cambio de filosofía en el diseño. En los años 70 (era CISC), se buscaba cerrar la "brecha semántica" entre el hardware y los lenguajes de alto nivel mediante instrucciones complejas interpretadas por microcódigo.
Sin embargo, en los años 80, diseñadores como David Patterson se dieron cuenta de que diseñar instrucciones que pudieran emitirse rápidamente era la clave para el rendimiento. Se priorizó maximizar la tasa de emisión de instrucciones sobre la complejidad de las mismas. Además, al eliminar el nivel de interpretación, se lograba mayor velocidad.

Diferencias Clave:

1. Ejecución: En RISC, las instrucciones se ejecutan directamente por hardware, mientras que en CISC (tradicional) se utiliza una unidad de control interpretada por microinstrucciones

2. Acceso a Memoria: En RISC, solo las instrucciones de carga (LOAD) y almacenamiento (STORE) hacen referencia a la memoria; el resto opera solo en registros. En CISC, las operaciones aritméticas pueden acceder directamente a memoria.

Cantidad y Complejidad de Instrucciones: RISC tiene un número pequeño de instrucciones (alrededor de 50) y busca formatos regulares/fijos fáciles de decodificar. CISC tiene cientos de instrucciones de longitud variable.

2. ¿Qué significa que un procesador Intel moderno tenga “un núcleo RISC interno?

Significa que, aunque la arquitectura externa es CISC (para mantener la compatibilidad con versiones anteriores y el software existente), internamente el hardware funciona con principios RISC.
Desde el procesador 486, las CPU de Intel contienen un núcleo RISC que ejecuta instrucciones simples directamente, mientras que las instrucciones x86 complejas (CISC) se decodifican o traducen a operaciones simples tipo RISC para su ejecución. Esto permite un rendimiento competitivo manteniendo la compatibilidad.

PROGRAMADOR UNIVERSITARIO
LICENCIATURA EN INFORMÁTICA
INGENIERÍA EN INFORMÁTICA
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 8

PRADO, MATIAS SANTIAGO

3. Dado el siguiente código en pseudo-ensamblador, identificar qué versiones serían más propias de un CISC y cuáles de un RISC, justificando:

A) MOV AX, [X]
ADD AX, [Y]
MOV [Z], AX

Esta versión es típica de CISC porque permite que una instrucción aritmética (ADD) opere directamente con un operando que se encuentra en la memoria ([Y]).

B) LOAD R1, X
LOAD R2, Y
ADD R3, R1, R2
STORE Z, R3

Esta versión corresponde a RISC (arquitectura Load/Store). Cumple con el principio de que "sólo las instrucciones LOAD y STORE deben hacer referencia a la memoria" y todas las demás operan exclusivamente entre registros.

4. ¿Por qué favorece tener instrucciones simples y de longitud fija?

Favorece el rendimiento porque la decodificación de instrucciones es un límite crítico en la tasa de emisión (instruction issue rate).
Instrucciones regulares o de longitud fija, con pocos campos y formatos, hacen que sean fáciles de decodificar. Esto permite al procesador determinar rápidamente qué recursos necesita cada instrucción y emitirlas más rápido.

5. ¿Qué hace un procesador “superescalar”?

Un procesador superescalar es aquel que puede emitir múltiples instrucciones por ciclo de reloj.
Para lograr esto, dispone de múltiples unidades funcionales a las que entrega estas instrucciones simultáneamente dentro de un único pipeline. La idea base es que la etapa de emisión sea capaz de alimentar a las unidades de ejecución más rápido de lo que una sola unidad podría completar el trabajo.

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 8

PRADO, MATIAS SANTIAGO

6. Mencione las diferencias entre SIMD y procesadores vectoriales.

Ambos explotan el paralelismo a nivel de datos, pero difieren en su implementación:

- SIMD (Single Instruction Multiple Data): Consta de un gran número de procesadores pequeños e idénticos (como en una GPU) que ejecutan la misma instrucción sobre diferentes datos simultáneamente.
- Procesadores Vectoriales: Utilizan una única unidad funcional fuertemente segmentada (pipelined). Operan sobre "registros vectoriales" (conjuntos de datos cargados en una sola instrucción) alimentándolos a un sumador segmentado.

7. ¿Qué es el principio de localidad? ¿Cuál es la diferencia entre localidad espacial y temporal?

El principio de localidad establece que los programas tienden a reutilizar datos e instrucciones que han utilizado recientemente (regla del 90/10). Esto permite predecir accesos futuros basándose en el pasado reciente.

- Localidad Temporal: Establece que es probable que se vuelva a acceder en un futuro próximo a los elementos a los que se accedió recientemente.
- Localidad Espacial: Establece que es probable que se acceda a elementos cuyas direcciones de memoria están cerca de las que se acaban de referenciar.

8. Analizar este fragmento de código:

```
for i = 1 to 1000:
        X = A[i] + A[i+1]
```

a. ¿Qué tipo de localidad de aprovecha?

Se aprovechan ambas.

b. ¿Qué parte del código activa la localidad temporal? ¿Y la espacial?

- localidad Espacial: El acceso al arreglo A. Al acceder a A[i], es muy probable que A[i+1] esté en la misma línea de caché o bloque de memoria, aprovechando la cercanía de direcciones.
- Localidad Temporal:

1. La variable X y el contador i se acceden repetidamente en cada iteración.
2. El elemento A[i+1] de la iteración actual se convierte en A[i] en la siguiente iteración. Por lo tanto, el dato cargado como A[i+1] se vuelve a utilizar casi inmediatamente como A[i], cumpliendo la definición de acceso reciente en el futuro próximo.

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 8

PRADO, MATIAS SANTIAGO

9. ¿Qué significa "centrarse en el caso común"? De un ejemplo distinto al mencionado en la teoría.

Significa que, al hacer un compromiso de diseño, se debe favorecer el caso frecuente sobre el poco frecuente. Invertir recursos en acelerar lo que ocurre el 90% del tiempo tiene mucho más impacto en el rendimiento global que optimizar eventos raros.
Otro ejemplo distinto de la teoría es la Predicción de Saltos: asumir que un bucle se repetirá (caso común) y ejecutar especulativamente, en lugar de esperar la evaluación de la condición.

10. Si el 90% del tiempo de un procesador se dedica a decodificar instrucciones y solo el 10% a operar en la ALU:
a. ¿Dónde conviene invertir recursos de diseño?

Conviene invertir en la unidad de decodificación de instrucciones.

b. Justificar con el principio del caso común.

El principio dicta que el impacto de una mejora es mayor si ocurre con frecuencia. Si la decodificación ocupa el 90% del tiempo (el caso común), reducir este tiempo a la mitad mejoraría el rendimiento total en un 45%. Si optimizamos la ALU (que solo usa el 10%), incluso si la hacemos infinitamente rápida, la mejora total máxima sería solo del 10%.

11. ¿Qué es un multiprocesador? ¿Cuál es la diferencia fundamental entre un multiprocesador y una multicomputadora?

Un multiprocesador es una computadora paralela donde todas las CPU comparten una memoria común y un único espacio de direcciones virtuales.
Diferencia Fundamental:

- Imagen del Sistema: En el multiprocesador hay una única copia del sistema operativo, una única tabla de procesos y mapa de páginas para todas las CPU.
- En la multicomputadora (memoria distribuida), cada CPU tiene su propia memoria privada y su propia copia del sistema operativo

12. ¿Cómo funciona la comunicación entre CPU en una multicomputadora?

Dado que no pueden leer la memoria ajena, se comunican pasando mensajes a través de una red de interconexión.
Si la CPU A necesita datos de la CPU B:

1. CPU A debe enviar un mensaje de solicitud a través de la red.
2. CPU A suele bloquearse esperando respuesta.
3. El software en CPU B recibe el mensaje, busca el dato y envía un mensaje de respuesta.
4. CPU A recibe el dato y se desbloquea.
   Esto se realiza mediante primitivas de software como "Enviar" (Send) y "Recibir" (Receive).

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 8

PRADO, MATIAS SANTIAGO

13. Tiene que diseñar un sistema para procesar datos científicos con 2000 nodos.

a. ¿Conviene más un multiprocesador o una multicomputadora?

Conviene una multicomputadora.

b. ¿Qué problemas de programación aparecerán?

Será difícil de programar. El programador deberá gestionar explícitamente la comunicación (mensajes), dividir correctamente los datos y colocarlos en ubicaciones óptimas para minimizar el tráfico de red.

c. ¿Qué ventajas tendrá el diseño elegido?

La principal ventaja es la facilidad de construcción y el costo reducido en comparación con un multiprocesador masivo. Permite escalar a miles de nodos (como los 2000 del enunciado) de forma viable

14. ¿Por qué existen los sistemas híbridos y qué buscan resolver?

Existen para resolver el dilema de que los multiprocesadores son fáciles de programar, pero difíciles de construir, mientras que las multicomputadoras son fáciles de construir, pero difíciles de programar. Buscan combinar lo mejor de ambos mundos: sistemas que sean escalables (fáciles de construir y agregar CPUs) y fáciles de programar (ofreciendo alguna abstracción de memoria compartida).

15. Definir DSM y explicar cómo simula memoria compartida.

DSM (Distributed Shared Memory - Memoria Compartida Distribuida) es un enfoque donde el sistema operativo simula memoria compartida sobre un hardware de multicomputadora.
Funcionamiento:

1. Se proporciona un único espacio de direcciones virtuales paginado compartido.
2. Cada página reside físicamente en una de las memorias locales de los nodos.
3. Cuando una CPU intenta acceder a una página que no tiene localmente, se produce un "fallo de página" (trap).
4. El SO localiza la página, solicita al nodo dueño que la envíe por la red y la asigna localmente, reiniciando la instrucción.
   Para el usuario, parece un multiprocesador real.

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 8

PRADO, MATIAS SANTIAGO

16. Utilizando lo visto en clase, analizar las ventajas y desventajas de implementar memoria compartida: a. en hardware.
b. en el sistema operativo.
c. en tiempo de ejecución (ej. Linda)

a)

- Ventaja: Alto rendimiento, acceso directo LOAD/STORE, transparencia total, modelo de programación sencillo (una sola imagen de SO).
- Desventaja: Difícil y costoso de construir para muchas CPUs.

b)

- Ventaja: Permite usar hardware de multicomputadora (barato y escalable) con una abstracción de memoria compartida para el usuario.
- Desventaja: Menor rendimiento debido a la gestión de fallos de página y tráfico de red gestionado por software (traps).

c)

- Ventaja: Implementación por software (bibliotecas/compilador) sin necesidad de soporte de hardware o SO especial. Modelo simple (tuplas).
- Desventaja: Requiere usar primitivas específicas (out, in, rd) en lugar de lectura/escritura de memoria convencional, lo que cambia el paradigma de programación.

17. Dado cada caso, clasificar en SISD, SIMD, MISD o MIMD:
a. Una GPU ejecutando la misma instrucción sobre miles de píxeles.
b. Un servidor con 4 CPU, cada una ejecutando procesos distintos.
c. Un microcontrolador simple ejecutando un solo hilo.
d. Un pipeline donde varias etapas procesan el mismo dato.

a)

Una unidad de control emite una instrucción que ejecutan múltiples ALUs sobre diferentes datos.

b)

Múltiples CPUs independientes funcionando como parte de un sistema mayor.

c)

La máquina clásica secuencial de Von Neumann.

d)

MISD (Multiple Instruction, Single Data). Aunque es una categoría rara, la teoría menciona explícitamente: "Algunas personas consideran que las máquinas con segmentación (pipeline) son MISD".

18. Dado este listado, clasificar cada uno según UMA / NUMA / COMA / NORMA:
a. Un servidor típico de 2 CPU que comparten RAM física.
b. Un clúster de 50 PCs conectados por Ethernet.
c. Una supercomputadora con miles de nodos y red de interconexión dedicada.
d. Un sistema donde la memoria principal se usa como caché.

a)

UMA (Uniform Memory Access). Cada CPU tiene el mismo tiempo de acceso a la memoria común.

b)

NORMA (NO Remote Memory Access). Es una multicomputadora tipo COW/NOW donde no hay acceso directo a memoria remota.

c)

NORMA (tipo MPP). Procesadores masivamente paralelos con red propietaria, sin acceso directo a memoria remota a nivel de instrucción.

d)

COMA (Cache Only Memory Access). La memoria principal se utiliza como caché en lugar de almacenamiento estático.

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II
TRABAJO PRÁCTICO Nº 8

PRADO, MATIAS SANTIAGO

19. Explicar la diferencia entre MPP y COW/NO

Ambas son multicomputadoras (NORMA), pero difieren en su construcción y costo:

- MPP (Massively Parallel Processors): Son supercomputadoras costosas, construidas como un sistema integrado con una red de interconexión propietaria (patentada) de alta velocidad y CPU estrechamente acopladas.
- COW/NOW (Cluster/Network of Workstations): Constan de computadoras estándar (PCs, servidores) conectadas mediante tecnología comercial disponible en el mercado (como Ethernet). Son una fracción del precio de un MPP.