Aspectos Avanzados de Arquitectura de Computadoras y Taller de Arquitectura de Computadoras

Prueba escrita correspondiente al curso 2011

- **Indique su nombre completo y número de cédula en cada hoja.**
- **Numere todas las hojas e indique el total de hojas en la primera.**
- **Escriba las hojas de un solo lado.**
- **Solo se responderán dudas de letras. No se responderán preguntas en los últimos 30 minutos de la prueba.**
- **La prueba es individual y sin material.**
- **Duración de la prueba : 2 horas y 30 minutos.**
- **Todas las preguntas tienen el mismo puntaje y el mínimo de aprobación es un un 60%.**

Ejercicio 1

Explique cuál es el objetivo de un controlador de DMA y cómo lo programa la CPU para transferir un bloque de datos de un dispositivo a memoria.

Respuesta Ejercicio 1

Ser una opción a la E/S manejada por interrupciones, sin la intervención activa de la CPU. Con DMA se permite realizar una transferencia de E/S, mientras que la CPU realiza otras tareas útiles.

Cuando se necesita hacer una transferencia de E/S, la CPU programa al dispositivo DMA con los siguientes parámetros:

- Configura el tipo de operación (Lectura/Escritura)
- Dirección del dispositivo que quiere transmitir los datos
- Dirección de comienzo del bloque de memoria
- Cantidad de datos a transferir

Una vez finalizada la transferencia, el DMA interrumpe a la CPU para indicar el final de la operación de E/S.

Otra alternativa es el “Robo” de ciclos del DMA, en el cual:

- Por cada palabra transferida, el DMA se adueña del bus
- No es una interrupción (no hay que cambiar de contexto)
- La CPU queda suspendida si necesita acceder al bus
- Puede enlentecer la CPU, pero no tanto como si la CPU tuviera que hacer la transferencia

Ejercicio 2

Explique en qué consiste y cuál es el uso de un TSS en la arquitectura IA-32 de Intel.

Respuesta Ejercicio 2

El Task State Segment (TSS) es una estructura que permite mantener los datos propios de ejecución de una tarea. Se guardan datos importantes del estado de la task como son el IP, los segmentos de código, de datos, de stack, el puntero de la pila, etc. Estos segmentos pueden ser en realidad referencias a la LDT, si tienen el bit de ese descriptor encendido, o de la GDT en su defecto.

Ejercicio 3 (sólo TAC)

Considere la siguiente instrucción en lenguaje ensamblador IA-32 de Intel: `add ax, es:[bx].`

Explique detalladamente cómo se obtienen los operandos para la suma cuando la CPU se encuentra operando en modo Protegido. Asuma que se accede a la gdt.

Respuesta Ejercicio 3

Aquí tenemos una operación de suma registro-memoria, de manera que la parte de registros es transparente, pero debemos obtener el operando que está en es:[bx].

Para obtener la dirección del segundo operando, lo primero que tenemos que obtener es el valor del segmento ES. Este es uno de los segmentos que podemos definir al momento de crear el TSS, y el cual apuntará a un descriptor de segmento en la GDT. En ese descriptor están las direcciones de comienzo y fin de dicho segmento. Luego, sumándole bx a la dirección base obtendremos la dirección donde se encuentra la palabra que buscamos.

Ejercicio 3 (sólo AAAC)

Dado el siguiente programa y un pipeline donde la penalización por saltos es de 5 ciclos.

```
main:
  mov $8, 0x10000000
  mov $9, 1000
  mov $10, 5
loop:
  lw $11, ($8+$9)
  addu $12,$11,$10
  sw $12, ($8+$9)
  addi $9,$9,-4
  bgtz $9,loop
fin:
  ret
```

a) Calcule aproximadamente los ciclos de espera necesarios por hazard de control.

b) Aplique la técnica de loop unrolling en el programa dado para aumentar el rendimiento en CPI en al menos un 30%.

Respuesta Ejercicio 3

a) Para resolver este ejercicio tenemos que asumir que usamos un predictor NotTaken, de manera que siempre predice no tomar el salto, y comenzar con la carga de la siguiente instrucción al pipeline.

El bucle se ejecuta 250 veces, comenzando en 1000 y se detiene cuando llega a cero, en pasos de a 4. En los primeros 249 se tendrá una penalización, y en el último se saldrá del loop. Entonces tendremos 1245 ciclos de espera.

b) Calcularemos el CPI total viejo y el nuevo.

CPIold = 5.250.CPI + 249.5

La cantidad de instrucciones que se ejecutan por bucle mas los ciclos de espera.

CPInew = 4.250.CPI + y.CPI + y.5

La variable y es la nueva cantidad de bucles, aunque varie la cantidad de bucles las demas instrucciones se siguen ejecutando 250 veces por el loop unrolling que copia las mismas varias veces, menos el salto que se ejecuta y veces.

Queremos lograr que CPInew <= 0,7 CPIold

Asumimos que el CPI de las instrucciones es 1 dado que es una máquina con pipeline y solo estamos considerando la penalización por salto.

1000 + 6.y <= 0,7.2495

despejando

y<= 124,4 es decir tienen que haber menos de 124 bucles.

Repitiendo el código por lo menos 3 veces seria suficiente. También hay que tener en cuenta que el programa funcione bien.

Ejercicio 4

a) Compare las técnicas de micro-programación vertical y horizontal.

b) ¿Cuál se usa en la microarquitectura vista en el curso (MIC-1)? Justifique.

c) Una unidad de control de microprograma contiene 1024 palabras de 100 bits cada una. Si solamente se utilizan 120 combinaciones de bits diferentes, ¿cuántos bits pueden ahorrarse utilizando nanoprogramación?

Respuestas Ejercicio 4

a) **Microprogramación horizontal**

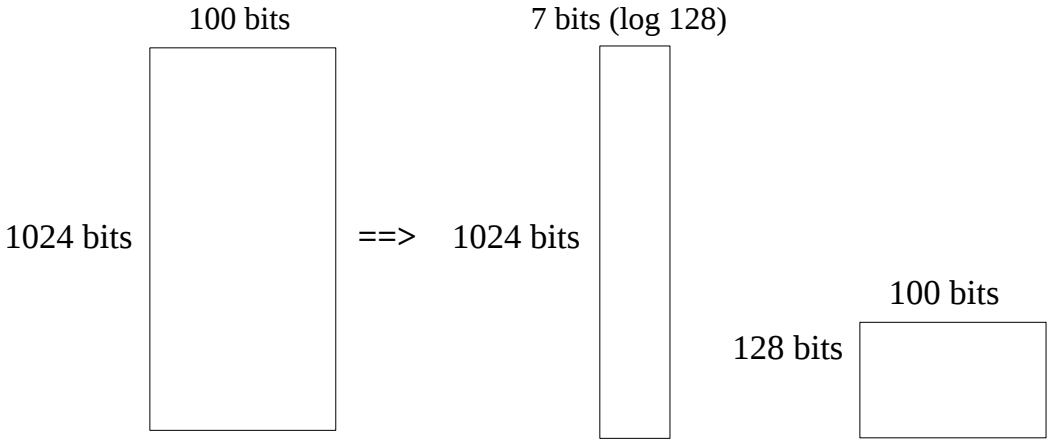
Cada microinstrucción especifica las señales de control sin codificar (muchas señales se pueden generar en paralelo). Esto lleva a tener una palabra de microcódigo ancha, pero con posibilidad de buen paralelismo, debido a la poca codificación de la información de control.

Microprogramación vertical

Se eligen algunas combinaciones de señales de control y se asigna un código a cada combinación. Esto lleva a tener palabras de microprograma mas angostas que en el paradigma horizontal, con log n bits combinaciones de señales de control codificadas. Esto a su vez lleva a que la capacidad sea nas limitada para expresar paralelismo, y también se requiere un decodificador externo para decodificar y manipular las señales de control.

b) MIC-1 es híbrido, ya que presenta algunos bits mapeados directamente a la operación, pero también tiene otros grupos de señales combinados, por ejemplo las operaciones de la ALU que se tienen 2 bits.

c) La disposición de memoria original es como sigue:



Se pasaría de una memoria de 102400 bits, a tener 1024*7+128*100 = 19968.

El ahorro en memoria será de 1024*100-19968 = 82432 bits

Ejercicio 5

Se considera una máquina sin coprocesador matemático que debe emular las operaciones de PF con secuencias de operaciones enteras. Su ciclo de reloj es de 10 ns. La siguiente tabla muestra las frecuencias relativas de las distintas operaciones de PF, y el número de instrucciones enteras necesarias para emular cada operación de PF.

Operación PF	**Frecuencia**	**N° de Instr. enteras p/ emularla**
Suma	50%	6
Multiplicación	40%	10
División	10%	20

Se agrega un coprocesador matemático, que elimina la necesidad de la emulación. Sin embargo, este agregado aumenta el ciclo de reloj un 20%.

Considere una carga de 150 millones de instrucciones, 20% de las cuales son enteras y 80% de PF. Asuma que el CPI=1 para las instrucciones enteras (y CPI=1 para las instrucciones de PF ejecutadas por el coprocesador).

a) Calcule los MIPS de la máquina original y de la nueva máquina con coprocesador.
b) Calcule la aceleración lograda con el agregado del coprocesador.
c) Repita este cálculo si se pudiera agregar el coprocesador sin afectar la duración del ciclo de reloj.

Respuestas Ejercicio 5

a)

Sin coprocesador:

- tenemos 20% (o sea 30 millones) son operaciones enteras que consumen 1 ciclo cada una
- tenemos 80% (o sea 120 millones) son operaciones de punto flotante, que se reparten de esta manera (0.5x6 + 0.4x10 + 0.1x20) = (3+4+2) = o sea que en promedio lleva 9 instrucciones enteras, resolver una de PF, o sea 9 ciclos
- en total CPI = 0.2+0.8x9 = 7.4
- MIPS = $1/ (1 \times 10^{6}) \times 7.4 \times (10 \times 10^{-9}) = 1/0{,}074 = 13{,}5$

Con coprocesador:

- Todas las operaciones llevan un ciclo, ya sean enteras o punto flotante
- entonces tenemos que CPI = 1
- MIPS = $1 / (1 \times 10^{6}) \times 1 \times (12 \times 10^{-9}) = 1/ 0{,}012 = 83{,}4$

b) tiempo viejo/tiempo nuevo = 7.4x10/1x12 = 74/12 = 1 / (mips viejo / mips nuevo) = 6.16

c) Siguiendo el mismo razonamiento que en la parte b), pero tomando que el ciclo de la máquina con la mejora también sea 10 ns, sería una mejora de 7.4