Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (I/2016)

Solución Interrogación 3

Respuestas sin desarrollo o justificación no tendrán puntaje.

Pregunta 1

a) Un computador tiene una memoria caché de 16KB, con líneas de 32 bytes que almacenan 8 palabras, y un tiempo de acceso de 10ns. La memoria caché está conectada a la memoria principal mediante un bus capaz de transferir 8 bytes en 120ns. ¿Cuál es el hit-rate que debe tener la memoria caché para tener un tiempo de acceso promedio de 20ns? **(1 pto.)**
Solución: $20ns = 10ns + (1 - HR) \times 480ns \Rightarrow HR \approx 0{,}98$

b) Comente sobre las ventajas de tener una memoria caché *split* en vez de una *unified* conectada a la memoria de datos del computador básico. **(1 pto.)**
Solución: No existen ventajas, sólo desventajas, ya que una memoria caché split conectada a una memoria de datos desperdiciará la mitad de su capacidad en instrucciones.

c) Responda las siguientes preguntas considerando el siguiente código, que implementa la multiplicación entre una matriz **A** y un vector **b**:

```
for (i=0; i<96; i++ )
{
    for (j=0; j<96; j++ )
    {
        c[i] = c[i]+ A[i][j] * b[j];
    }
}
```

i. Determine la cantidad de *misses* y el *hit-rate* de la memoria caché, asumiendo que todos los elementos son números del tipo double y que la matriz **A** está almacenada en orden de filas. La memoria caché es *fully associative* y tiene capacidad infinita, con líneas de 64 bytes. Considere además que el vector **c** ha sido inicializado con ceros previamente y que no existen elementos de **A**, **b** y **c** almacenados en la caché. **(2 ptos.)**
Solución: Para el vector **c**, dado que las líneas son de 84 bytes, cada lectura cargará 8 elementos, por lo que se tendrán 12 *misses*. Usando este mismo argumento, se tendrán también 12 *misses* para el vector **b** y $96 \times 12 = 1152$ para la matriz **A**, lo que da un total de 1176 *misses*. Finalmente, dado que por iteración se realizan 4 accesos a memoria, se tienen en total $96 \times 96 \times 4 = 36864$ accesos, lo que implica que el *miss-rate* es $\frac{1176}{36864} = 0.0319 = 3.19\,\%$.

ii. Asuma ahora que la matriz **A** utiliza una memoria caché de 4KB con mapeo directo para ella sola, *i.e.*, **b** y **c** usan una memoria caché distinta a esta. Asumiendo que A[0][0] es mapeado a la primera

línea de la caché y que el tamaño de un bloque es de 64 bytes, indique los valores de **i** y **j**, para los cuales el bloque usado por A[95][95] sustituirá al bloque usado por A[i][j] en la caché, y la línea de la caché donde ocurrirá esto. **(2 ptos.)**
Solución: Dado que las líneas son de 64 bytes, cada fila de la matriz **A** utilizará $\frac{96\times 8}{64} = 12$ líneas. Además, como la memoria caché es de 4KB, esta tendrá $\frac{4096}{64} = 64$ líneas. Luego, la matriz completa utilizará exactamente $\frac{96*12}{64} = 18$ veces la capacidad completa de la memoria caché, lo que implica que el bloque que contiene a A[95][95] será asignado a la línea 63. Finalmente, la distancia entre el bloque que contiene A[95][95] y el que contiene al A[i][j] que será sustituido es de 12 líneas, lo que equivale a $5,\overline{3}$ filas completas de la matriz. Esto implica que el elemento que será sustituido por el bloque de A[95][95] en la caché será A[90][32].

Pregunta 2

a) En un computador con soporte para memoria virtual, ¿cómo se pueden implementar regiones de memoria protegidas y regiones de memoria compartidas? **(1 pto.)**
Solución: Las regiones de memoria compartida se pueden implementar asignando el mismo marco de memoria física a páginas de procesos distintos. Las regiones de memoria compartida se pueden implementar de manera trivial prohibiendo la asignación de los marcos que las contienen a cualquier página virtual.

b) Indique que eventos generan un cambio de contexto en el esquema de *cooperative scheduling*. **(1 pto.)**
Solución: El cambio de contexto puede darse luego de una cesión explícita por parte del proceso (*yield*), o tras a una petición de interacción con un dispositivo de I/O.

c) Considere un computador con un espacio direccionable virtual de 32 bits, espacio de direccionamiento físico de 30 bits y páginas de 8KB.

   I. Describa la composición interna de una dirección virtual. **(1 pto.)**
   Solución: Una dirección virtual se dividirá utilizando los 14 bits menos significativos para indicar el offset dentro de una página y los restantes 18 bits para obtener el mapeo desde página a marco.

   II. ¿Cuál es el máximo número de entradas válidas que pueden existir en una tabla de páginas? **(1 pto.)**
   Solución: Asumiendo que existe un único proceso, se necesita por lo menos un marco que almacene la tabla de páginas de este. Por lo tanto, la cantidad máxima de entradas válidas será $\frac{2^{30}}{2^{14}} - 1 = 2^{16} - 1 = 65535$.

d) En un computador con soporte para memoria virtual, ¿el registro PC almacena direcciones virtuales o físicas? Justifique su respuesta considerando el efecto de los cambios de contexto y del *swapping*. **(2 ptos.)**
Solución: Dado que un proceso sólo genera peticiones de memoria utilizando direcciones virtuales, es conveniente que el PC también almacene direcciones de este tipo. En caso contrario, existen numerosos problemas de consistencia de memoria que se pueden dar. Tomando como consideración los cambios de contexto y *swapping*, es muy posible que un proceso que vuelve a ejecutarse despues de haber estado en espera por una petición de I/O, no tengo su código en la misma ubicación física que antes, debido al swapping de memoria. Si el PC almacena la dirección física, este apuntará ahora a una dirección que no contiene necesariamente el código correcto. Por el contrario, si almacena direcciones virtuales, la transformación de estas a direcciones físicas siempre llevaran a la ubicación correcta del código, independiente si este cambió de ubicación física.

Pregunta 3

a) Estime el tiempo de ejecución de un programa que toma **N** instrucciones en el computador básico con *pipeline*, en base a las siguientes condiciones:

- El 50 % de las instrucciones realizan lecturas o escrituras en memoria.
- El 10 % de las instrucciones son de salto y en el 25 % de estas, el salto finalmente se realiza.
- En el 50 % de las ocasiones, el dato obtenido desde la memoria debe ser utilizado en la instrucción siguiente.

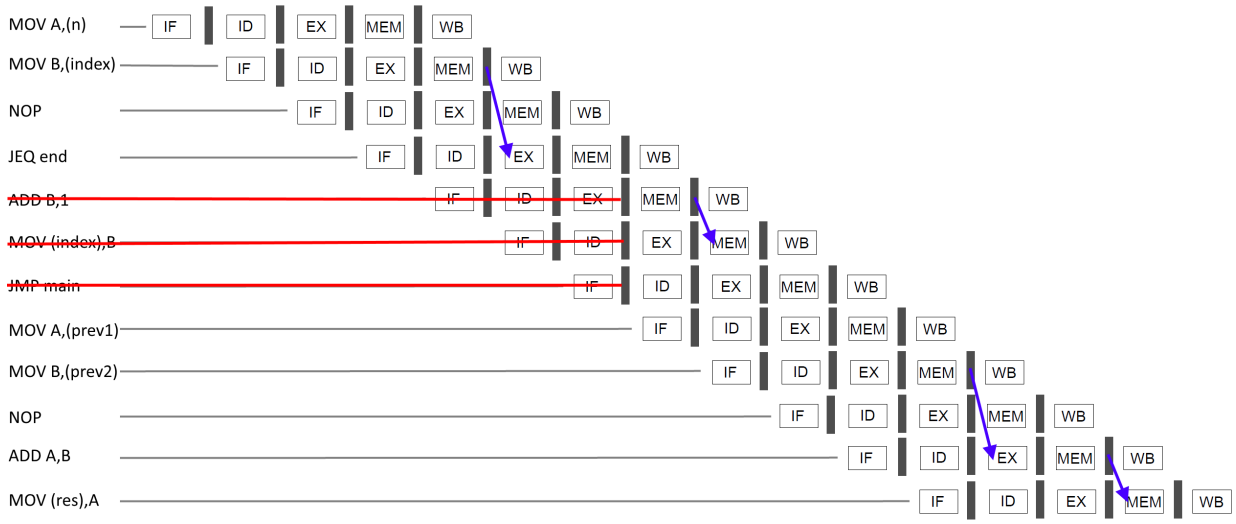
Cualquier supuesto sobre la solución debe quedar claramente explicado. **(1 pto.)**
Solución: Si se obvían las restricciones y sólo se considera la cantidad de instrucciones ejecutadas por el programa, el tiempo de ejecución será de N+4 ciclos. Si se consideran en conjunto la primera y tercera condiciones, por cada una de las instrucciones afectadas, se deberá agregar una burbuja que retrasará la ejecución en un ciclo, lo que en la práctica para la totalidad del programa significa un retraso de $0{,}5 \times 0{,}5 \times N = 0{,}25N$ ciclos. Adicionalmente, si se considera la segunda restricción, se agregarán 3 ciclos por cada salto realizado, retrasando el programa $0{,}1 \times 0{,}25 \times 3 \times N = 0{,}075N$ ciclos. Tomando todo esto en consideración, el tiempo total de ejecución del programa será de $N + 0{,}25N + 0{,}075N + 4 = 1{,}325N + 4$.

b) Determine el número de ciclos que se demora el siguiente código, detallando en un diagrama los estados del pipeline por instrucción. El pipeline tiene forwarding entre todas sus etapas, el manejo de stalling es por software (instrucción NOP) y predicción de salto asumiendo que no ocurre. Indique en el diagrama cuando ocurre forwarding, stalling y flushing. **(2 ptos.)**

```
DATA
    n     1
    index 1
    prev1 0
    prev2 1
    res   0

CODE
    main:
        MOV A, (n)
        MOV B, (index)
        JEQ end
        ADD B, 1
        MOV (index), B
        JMP main
    end:
        MOV A, (prev1)
        MOV B, (prev2)
        ADD A, B
        MOV (res), A
```

Solución: En base al diagrama presentado a continuación, el programa toma 16 ciclos en ejecutarse.



c) Diseñe un computador con *pipeline* de al menos 5 etapas, donde debido a restricciones del hardware, la etapa MEM debe ejecutarse antes que la etapa EX. El computador debe soportar las mismas funcionalidades que el computador básico con *pipeline*. **(3 ptos.)**

Solución: La clave para solucionar este ejercicio consiste en evitar que los saltos mal predecidos afecten el contenido de la memoria, que complica la aplicación de una operación de flushing. En el computador básico con pipeline, esto se evita ubicando la etapa MEM después de EX, dado que la condición de salto se genera en la etapa EX y se evalua en la etapa MEM. Teniendo esto en consideración, una posible solución consiste en un computador con un pipeline de 6 etapas: IF, ID, JMP, MEM, EX, WB, donde la etapa JMP contiene un restador que genera la condición de salto. Luego, tal como en el computador básico con pipeline, en la etapa MEM se evalua la condición, evitando la escritura en memoria de las instrucciones siguientes.