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

IIC2343 – Arquitectura de Computadores (I/2017)

Tarea 4

Fecha de entrega: jueves 1 de junio de 2016 a las 23:59 horas
Ayudante encargado: Patricio Díaz (pndiaz1@uc.cl)

Jerarquía de memoria

En esta tarea tendrán la oportunidad de experimentar con el diseño y evaluación de una memoria caché. Mas específicamente, deberán calcular métricas de rendimiento de una jerarquía de memoria de 2 niveles (caché y memoria principal), cuando se ve enfrentada a la ejecución de un programa escrito en el assembly de la tarea 2.

La tarea desarrollada debe recibir como entrada a través de la consola, el nombre de un archivo de texto que contiene el código a ejecutar. Como salida, la tarea debe entregar el *hit-rate* y el *write-rate* correspondientes al código, donde el *write-rate* se define como la cantidad de escrituras realizadas efectivamente en la memoria de datos dividida por la cantidad de escrituras en esta memoria que se realizarían si no existiese la caché.

La jerarquía de memoria que deberá simular, tiene las siguientes características:

- Caché: split, 64B totales.
- Memoria principal: 256B.
- Tamaño de línea: 4B.
- Función de correspondencia: 2-way associative.
- Política de sustitución: LRU.
- Política de escritura: *write-back*.

Las instrucciones soportadas y el formato archivo del texto de entrada, serán iguales a los de la tarea 2. Considere además que cada palabra de la memoria de instrucciones utiliza 16 bits (8 bits opcode, 8 bits literal). El archivo generado por el proceso deberá presentar los resultados de la siguiente manera:

```
HR=0.7
WR=0.2
```

Es importante notar que los números utilizados para indicar tanto hit-rate y como write-rate pueden tener precisión arbitraria.

Implementación

Para la realización de esta tarea deberán trabajar en el lenguaje Python 3.x, sin ocupar bibliotecas externas a las del lenguaje. Para ejecutar la tarea, se deberá entregar por línea de comandos el nombre del archivo a procesar:

```
$ python {nº alumno}.py archivo.txt
```

Luego de esto, el programa deberá escribir en el archivo {nº alumno}.txt los resultados requeridos.

Entrega y evaluación

La tarea se debe realizar de **manera individual** y la entrega se realizará mediante un cuestionario a través del sitio del curso. El formato de entrega debe consistir en un único archivo .py, que lleve como nombre el número de alumno, y que contenga sólo el código fuente. No incluya en su entrega contenido relacionado con el entorno de desarrollo utilizado. El no cumplir este formato de entrega implicará un descuento de **1.0** punto en la nota final de la tarea. Tareas incompletas serán evaluadas con nota **1.0**. En caso de atraso, se aplicará un descuento de **1.0** punto por cada 6 horas o fracción.

Finalmente, todas las tareas serán analizadas electrónicamente por posibles copias o plagio. Por "copia o plagio", se entiende incluir en el trabajo presentado como propio, partes hechas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente. En caso que se detecte alguna situación de este tipo, los antecedentes serán enviados a la Dirección de Pregrado de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario.

Anexo: instrucciones soportadas

A continuación se presenta una lista completa de las instrucciones del assembly que deben estar soportadas:

Instrucción	Operandos	Operación	Ejemplo de uso
MOV	A,B	A=B	-
	B,A	B=A	-
	A,Lit	A=Lit	MOV A,15
	B,Lit	B=Lit	MOV B,15
	A,(Dir)	A=Mem[Dir]	MOV A,(var1)
	B,(Dir)	B=Mem[Dir]	MOV B,(var2)
	(Dir),A	Mem[Dir]=A	MOV (var1),A
	(Dir),B	Mem[Dir]=B	MOV (var2),B
	A,(B)	A=Mem[B]	-
	B,(B)	B=Mem[B]	-
	(B),A	Mem[B]=A	-
ADD	A,B	A=A+B	-
	B,A	B=A+B	-
	A,Lit	A=A+Lit	ADD A,5
	A,(Dir)	A=A+Mem[Dir]	ADD A,(var1)
	A,(B)	A=A+Mem[B]	-
	(Dir)	Mem[Dir]=A+B	ADD (var1)
SUB	A,B	A=A-B	-
	B,A	B=A-B	-
	A,Lit	A=A-Lit	SUB A, 2
	A,(Dir)	A=A-Mem[Dir]	SUB A,(var1)
	A,(B)	A=A-Mem[B]	-
	(Dir)	Mem[Dir]=A-B	SUB (var1)
AND	A,B	A=A and B	-
	B,A	B=A and B	-
	A,Lit	A=A and Lit	AND A,15
	A,(Dir)	A=A and Mem[Dir]	AND A,(var1)
	A,(B)	A=A and Mem[B]	-
	(Dir)	Mem[Dir]=A and B	AND (var1)
OR	A,B	A=A or B	-
	B,A	B=A or B	-
	A,Lit	A=A or Lit	OR A,5
	A,(Dir)	A=A or Mem[Dir]	OR A,(var1)
	A,(B)	A=A or Mem[B]	-
	(Dir)	Mem[Dir]=A or B	OR (var1)
NOT	A,A	A=notA	-
	B,A	B=notA	-
	(Dir)	Mem[Dir]=not A	NOT (var1)

Instrucción	Operandos	Operación	Ejemplo de uso
XOR	A,B	A=A xor B	-
	B,A	B=A xor B	-
	A,Lit	A=A xor Lit	XOR A,15
	A,(Dir)	A=A xor Mem[Dir]	XOR A,(var1)
	A,(B)	A=A xor Mem[B]	-
	(Dir)	Mem[Dir]=A xor B	XOR (var1)
SHL	A,A	A=shift left A	-
	B,A	B=shift left A	-
	(Dir)	Mem[Dir]=shift left A	SHL (var1)
SHR	A,A	A=shift right A	-
	B,A	B=shift right A	-
	(Dir)	Mem[Dir]=shift right A	SHR (var1)
INC	B	B=B+1	-
CMP	A,B	A-B	
	A,Lit	A-Lit	CMP A,0
JMP	Dir	PC = Dir	JMP end
JEQ	Dir	PC = Dir	JEQ label
JNE	Dir	PC = Dir	JNE label
JGT	Dir	PC = Dir	JGT label
JLT	Dir	PC = Dir	JLT label
JGE	Dir	PC = Dir	JGE label
JLE	Dir	PC = Dir	JLE label