CMU 18-447 Introduction to Computer Architecture, Spring 2015
HW 4: SIMD, VLIW, Static Scheduling, Caching, Virtual Memory

Instructor: Prof. Onur Mutlu
TAs: Kevin Chang, Rachata Ausavarungnirun, Albert Cho, Jeremie Kim, Clement Loh

Assigned: Wed., 2/25, 2015
Due: **Wed., 3/18, 2015 (Midnight)**
Handin: autolab

1 Vector Processing [15 points]

Consider the following piece of code:

```
for (i = 0; i < 100; i ++)
  A[i] = ((B[i] * C[i]) + D[i])/2;
```

(a) Translate this code into assembly language using the following instructions in the ISA (note the number of cycles each instruction takes is shown next to each instruction):

Opcode	Operands	Number of Cycles	Description
LEA	Ri, X	1	Ri ← address of X
LD	Ri, Rj, Rk	11	Ri ← MEM[Rj + Rk]
ST	Ri, Rj, Rk	11	MEM[Rj + Rk] ← Ri
MOVI	Ri, Imm	1	Ri ← Imm
MUL	Ri, Rj, Rk	6	Ri ← Rj x Rk
ADD	Ri, Rj, Rk	4	Ri ← Rj + Rk
ADD	Ri, Rj, Imm	4	Ri ← Rj + Imm
RSHFA	Ri, Rj, amount	1	Ri ← RSHFA (Rj, amount)
BRcc	X	1	Branch to X based on condition codes

Assume one memory location is required to store each element of the array. Also assume that there are 8 registers (R0 to R7).

Condition codes are set after the execution of an arithmetic instruction. You can assume typically available condition codes such as zero, positive, and negative.

How many cycles does it take to execute the program?

(b) Now write Cray-like vector assembly code to perform this operation in the shortest time possible. Assume that there are 8 vector registers and the length of each vector register is 64. Use the following instructions in the vector ISA:

Opcode	Operands	Number of Cycles	Description
LD	Vst, #n	1	Vst ← n (Vst = Vector Stride Register)
LD	Vln, #n	1	Vln ← n (Vln = Vector Length Register)
VLD	Vi, X	11, pipelined	
VST	Vi, X	11, pipelined	
Vmul	Vi, Vj, Vk	6, pipelined	
Vadd	Vi, Vj, Vk	4, pipelined	
Vrshfa	Vi, Vj, amount	1	

How many cycles does it take to execute the program on the following processors? Assume that memory is 16-way interleaved.

(i) Vector processor without chaining, 1 port to memory (1 load or store per cycle).

(ii) Vector processor with chaining, 1 port to memory.

(iii) Vector processor with chaining, 2 read ports and 1 write port to memory.

2 VLIW [15 points]

You are using a tool that transforms machine code that is written for the MIPS ISA to code in a VLIW ISA. The VLIW ISA is identical to MIPS except that multiple instructions can be grouped together into one *VLIW instruction*. Up to N MIPS instructions can be grouped together (N is the machine width, which depends on the particular machine). The transformation tool can reorder MIPS instructions to fill VLIW instructions, as long as loads and stores are not reordered relative to each other (however, independent loads and stores can be placed in the same VLIW instruction). You give the tool the following MIPS program (we have numbered the instructions for reference below):

```
(01) lw   $t0 ← 0($a0)
(02) lw   $t2 ← 8($a0)
(03) lw   $t1 ← 4($a0)
(04) add  $t6 ← $t0, $t1
(05) lw   $t3 ← 12($a0)
(06) sub  $t7 ← $t1, $t2
(07) lw   $t4 ← 16($a0)
(08) lw   $t5 ← 20($a0)
(09) srlv $s2 ← $t6, $t7
(10) sub  $s1 ← $t4, $t5
(11) add  $s0 ← $t3, $t4
(12) sllv $s4 ← $t7, $s1
(13) srlv $s3 ← $t6, $s0
(14) sllv $s5 ← $s0, $s1
(15) add  $s6 ← $s3, $s4
(16) add  $s7 ← $s4, $s6
(17) srlv $t0 ← $s6, $s7
(18) srlv $t1 ← $t0, $s7
```

(a) Draw the dataflow graph of the program Represent instructions as numbered nodes (01 through 18), and flow dependences as directed edges (arrows).

(b) When you run the tool with its settings targeted for a particular VLIW machine, you find that the resulting VLIW code has 9 VLIW instructions. What minimum value of N must the target VLIW machine have?

(c) Write the MIPS instruction numbers (from the code above) corresponding to each VLIW instruction, for this value of N. When there is more than one MIPS instruction that could be placed into a VLIW instruction, choose the instruction that comes earliest in the original MIPS program.

	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.
VLIW Instruction 1:										
VLIW Instruction 2:										
VLIW Instruction 3:										
VLIW Instruction 4:										
VLIW Instruction 5:										
VLIW Instruction 6:										
VLIW Instruction 7:										
VLIW Instruction 8:										
VLIW Instruction 9:										

(d) You find that the code is still not fast enough when it runs on the VLIW machine, so you contact the VLIW machine vendor to buy a machine with a larger machine width N. What minimum value of N

would yield the maximum possible performance (i.e., the fewest VLIW instructions), assuming that all MIPS instructions (and thus VLIW instructions) complete with the same fixed latency and assuming no cache misses?

(e) Write the MIPS instruction numbers corresponding to each VLIW instruction, for this optimal value of N. Again, as in part (c) above, pack instructions such that when more than one instruction can be placed in a given VLIW instruction, the instruction that comes first in the original MIPS code is chosen.

	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.
VLIW Instruction 1:										
VLIW Instruction 2:										
VLIW Instruction 3:										
VLIW Instruction 4:										
VLIW Instruction 5:										
VLIW Instruction 6:										
VLIW Instruction 7:										
VLIW Instruction 8:										
VLIW Instruction 9:										

(f) A competing processor design company builds an in-order superscalar processor with the same machine width N as the width you found in part (b) above. The machine has the same clock frequency as the VLIW processor. When you run the original MIPS program on this machine, you find that it executes slower than the corresponding VLIW program on the VLIW machine in part (b). Why could this be the case?

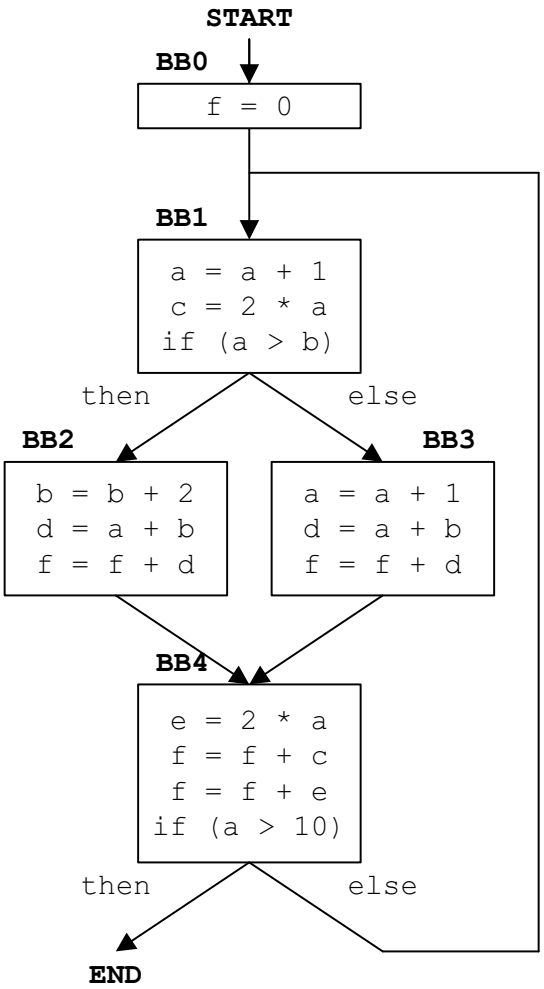
(g) When you run some other program on this superscalar machine, you find it runs faster than the corresponding VLIW program on the VLIW machine. Why could this be the case?

3 Code Optimizations [20 points]

Assume an **in-order**, **non-pipelined** machine. The ISA for this machine consists of only the following five instructions that never generate exceptions. In this syntax, x is a register, whereas y and z are either registers or constants. Note that this machine is able to execute branches and jumps **instantaneously**.

Instruction Name	Syntax	Cycles
Move	$x = y$	1
Addition	$x = y + z$	2
Multiplication	$x = y * z$	5
Branch-If-Greater-Than	`if (`y `>` z`)`	0
Jump	–	0

Consider the following assembly program that is written in the machine's ISA. For any initial values of registers a and b, the purpose of the program is to compute the final value and store it in register f. After the program terminates, it is considered to have executed correctly if and only if the values stored in registers a, b, and f are correct.



(a) How many cycles does the machine take to execute the assembly program, when a and b are initialized to 0 and 1, respectively?

You decide to make simple optimizations to the program to reduce its execution time. The optimizations involve only **removing**, **modifying**, and/or **moving** some of the already existing instructions. However, there are two restrictions: you may **not** move instructions out of the loop and you may **not** add completely new instructions.

(b) Show the optimized assembly program.

(c) How many cycles does the machine take to execute the optimized assembly program, when a and b are initialized to 0 and 1?

After learning about *superblocks*, you decide to optimize the program even further to reduce its execution time. In order to form the superblock(s), assume that you run the program once beforehand when a and b are initialized to 0 and 1. During this profile run, if a branch is biased in either direction by more than **60%**, it is included in the superblock(s). However, there are two restrictions: you may **not** move instructions out of the loop and you may **not** unroll the loop.

(d) Show the superblock-optimized assembly program. **Circle** the superblock(s).

(e) How many cycles does the machine take to execute the superblock-optimized assembly program, when a and b are initialized to 0 and 1?

(f) If you had used *traces* to optimize the program instead of superblocks, would the execution time increase, decrease, or stay the same compared to (e)? Choose one and explain briefly why.

(g) If you had used *hyperblocks* to optimize the program instead of superblocks, would the execution time increase, decrease, or stay the same compared to (e)? Choose one and explain briefly why.

4 Caching [15 points]

Below, we have given you four different sequences of addresses generated by a program running on a processor with a data cache. Cache hit ratio for each sequence is also shown below. Assuming that the cache is initially empty at the beginning of each sequence, find out the following parameters of the processor's data cache:

- Associativity (1, 2 or 4 ways)
- Block size (1, 2, 4, 8, 16, or 32 bytes)
- Total cache size (256 B, or 512 B)
- Replacement policy (LRU or FIFO)

Assumptions: all memory accesses are one byte accesses. All addresses are byte addresses.

Sequence No.	Address Sequence	Hit Ratio
1	0, 2, 4, 8, 16, 32	0.33
2	0, 512, 1024, 1536, 2048, 1536, 1024, 512, 0	0.33
3	0, 64, 128, 256, 512, 256, 128, 64, 0	0.33
4	0, 512, 1024, 0, 1536, 0, 2048, 512	0.25

5 Virtual Memory [10 points]

An ISA supports an 8-bit, byte-addressable virtual address space. The corresponding physical memory has only 128 bytes. Each page contains 16 bytes. A simple, one-level translation scheme is used and the page table resides in physical memory. The initial contents of the frames of physical memory are shown below.

Frame Number	Frame Contents
0	Empty
1	Page 13
2	Page 5
3	Page 2
4	Empty
5	Page 0
6	Empty
7	Page Table

A three-entry translation lookaside buffer that uses Least Recently-Used (LRU) replacement is added to this system. Initially, this TLB contains the entries for pages 0, 2, and 13. For the following sequence of references, put a circle around those that generate a TLB hit and put a rectangle around those that generate a page fault. What is the hit rate of the TLB for this sequence of references? (Note: LRU policy is used to select pages for replacement in physical memory.)

References (to pages): 0, 13, 5, 2, 14, 14, 13, 6, 6, 13, 15, 14, 15, 13, 4, 3.

(a) At the end of this sequence, what three entries are contained in the TLB?

(b) What are the contents of the 8 physical frames?

6 Page Table Bits [5 points]

(a) What is the purpose of the "reference" or "accessed" bit in a page table entry?

(b) Describe what you would do if you did not have a reference bit in the PTE. Justify your reasoning and/or design choice.

(c) What is the purpose of the dirty or modified bit in a page table entry?

(d) Describe what you would do if you did not have a modified bit in the PTE. Justify your reasoning and/or design choice.

7 Virtual Memory - Optional(From past exam..)

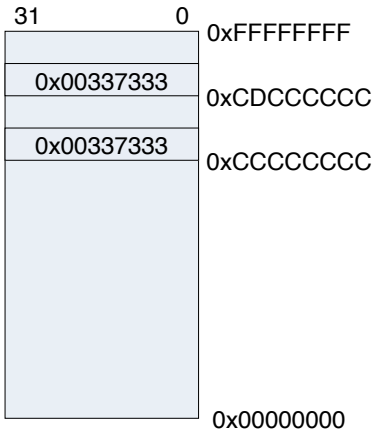
A dedicated computer architecture student (like you) bought a 32-bit processor that implements paging-based virtual memory using a single-level page table. Being a careful person, she also read the manual and learnt that

- A page table entry (PTE) is 4-bytes in size.
- A PTE stores the physical page number in the least-significant bits. Unused bits are zeroed out.
- The page table base address (PTBA) is page-aligned.
- If an instruction attempts to modify a PTE, an exception is raised.

However, the manual did not mention the page size and the PTBA. The dedicated student then wrote the following program to find out the missing information.

char *ptr = 0xCCCCCCCC;
ptr = 0x00337333;

The code ran with no exceptions. The following figure shows relevant locations of the physical memory **after** execution.



Using these results, what is the PTBA of this machine?

What is the page size (in bytes) of this machine? Write your answer in the form 2^n.