CMU 18-447 Introduction to Computer Architecture, Spring 2015
HW 1 Solutions: Instruction Set Architecture (ISA)

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

1 The SPIM Simulator [5 points]

There isn't a solution per se to this. The expectation is that you are familiar with SPIM/XSPIM, understand its strengths and limitations, and are using it to debug your labs and homeworks.

2 Big versus Little Endian Addressing [5 points]

1.

3a	2b	fe	ca
0	1	2	3

2.

ca	fe	2b	3a
0	1	2	3

3 Instruction Set Architecture (ISA) [25 points]

Code size:
Each instruction has an opcode and a set of operands

- The opcode is always 1 byte (8 bits).
- All register operands are 1 byte (8 bits).
- All memory addresses are 2 bytes (16 bits).
- All data operands are 4 bytes (32 bits).
- All instructions are an integral number of bytes in length.

Memory Bandwidth:
Memory bandwidth consumed = amount of code transferred (code size) + amount of data transferred
Amount of data transferred = number of data references * 4 bytes

We will call the amount of code transferred as I-bytes and the amount of data transferred as D-bytes.

(a), (b)

Instruction Set Architecture	Opcode	Operands	I-bytes	D-bytes	Total Bytes
Zero-address					
	PUSH	B	3	4	
	PUSH	C	3	4	
	ADD		1		
	POP	A	3	4	
	PUSH	A	3	4	
	PUSH	C	3	4	
	ADD		1		
	POP	B	3	4	
	PUSH	A	3	4	
	PUSH	B	3	4	
	SUB		1		
	POP	D	3	4	
			30	36	66

Instruction Set Architecture	Opcode	Operands	I-bytes	D-bytes	Total Bytes
One-address					
	LOAD	B	3	4	
	ADD	C	3	4	
	STORE	A	3	4	
	ADD	C	3	4	
	STORE	B	3	4	
	LOAD	A	3	4	
	SUB	B	3	4	
	STORE	D	3	4	
			24	32	56
Two-address					
	SUB	A, A	5	12	
	ADD	A, B	5	12	
	ADD	A, C	5	12	
	SUB	B, B	5	12	
	ADD	B, A	5	12	
	ADD	B, C	5	12	
	SUB	D, D	5	12	
	ADD	D, A	5	12	
	SUB	D, B	5	12	
			45	108	153
Three-address					
Memory-Memory	ADD	A, B, C	7	12	
	ADD	B, A, C	7	12	
	SUB	D, A, B	7	12	
			21	36	57
Three-address					
Load-Store	LD	R1, B	4	4	
	LD	R2, C	4	4	
	ADD	R1, R1, R2	4		
	ST	R1, A	4	4	
	ADD	R3, R1, R2	4		
	ST	R3, B	4	4	
	SUB	R3, R1, R3	4		
	ST	R3, D	4	4	
			32	20	52

(c) The three-address memory-memory machine is the most efficient as measured by code size - 21 bytes.

(d) The three-address load-store machine is the most efficient as measured by total memory bandwidth consumption (amount of code transferred + amount of data transferred) - 52 bytes.

4 The MIPS ISA [40 points]

4.1 Warmup: Computing a Fibonacci Number [15 points]

NOTE: More than one correct solution exists, this is just one potential solution.

```
fib:
addi $sp, $sp, -16 // allocate stack space
sw   $16, 0($sp)   // save r16
add  $16, $4,  $0  // r16 for arg n
sw   $17, 4($sp)   // save r17
add  $17, $0,  $0  // r17 for a, init to 0
sw   $18, 8($sp)   // save r18
```

```
addi $18, $0,  1   // r18 for b, init to 1
sw   $31, 12($sp)  // save return address
add  $2,  $17, $18 // c = a + b

branch:
slti $3,  $16, 2   // use r3 as temp
bne  $3,  $0,  done
add  $2,  $17, $18 // c = a + b
add  $17, $18, $0  // a = b
add  $18, $2,  $0  // b = c
addi $16, $16, -1  // n = n - 1
j    branch

done:
lw   $31, 12($sp)  // restore r31
lw   $18, 8($sp)   // restore r18
lw   $17, 4($sp)   // restore r17
lw   $16, 0($sp)   // restore r16
addi $sp, $sp, 16  // restore stack pointer
jr   $31           // return to caller
```

4.2 MIPS Assembly for REP MOVSB [25 points]

Assume: \$1 = ECX, \$2 = ESI, \$3 = EDI

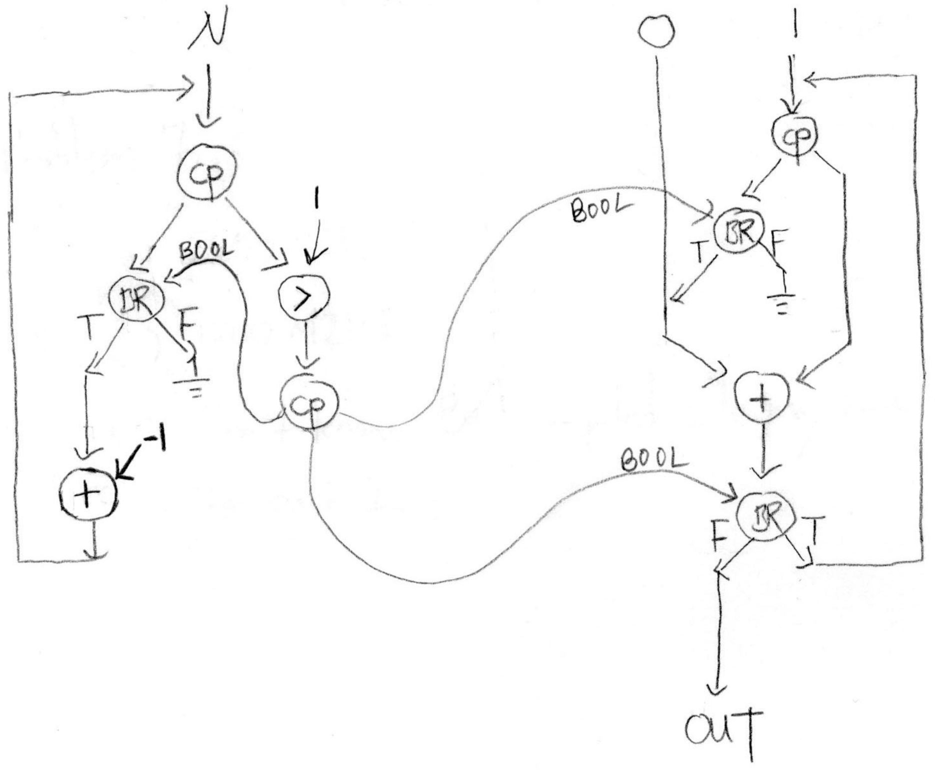
(a)
```
beq    $1, $0, AfterLoop          // If counter is zero, skip
CopyLoop:
lb     $4, 0($2)                  // Load 1 byte
sb     $4, 0($3)                  // Store 1 byte
addiu  $2, $2, 1                  // Increase source pointer by 1 byte
addiu  $3, $3, 1                  // Increase destination pointer by 1 byte
addiu  $1, $1, -1                 // Decrement counter
bne    $1, $0, CopyLoop           // If not zero, repeat
AfterLoop:
Following instructions
```

(b) The size of the MIPS assembly code is 4 bytes $\times$ 7 = 28 bytes, as compared to 2 bytes for x86 REP MOVSB.

(c) The count (value in ECX) is 0xfee1dead = 4276215469. Therefore, loop body is executed (4276215469 * 6) = 25657292814 times. Total instructions executed = 25657292814 + 1 (beq instruction outside of the loop) = 25657292815.

(d) The count (value in ECX) is 0x00000000 = 0. Therefore, loop body is executed 0 times. Total instructions executed = 1 (beq instruction outside of the loop).

5 Data Flow Programs [15 points]



6 DRAM Refresh [30 points]

(a) Capacity = 2^{60} and row size = 2^{13}, so there are 2^{47} rows.

(b) Because each row needs to be refreshed every 64ms, all rows need to be refreshed within 64ms. Thus, the number of refreshes = number of rows = 2^{47}.

(c) Given that we know the number of refreshes from (b), $P_{total} = N_{refresh} * P_{refresh}$. $P_{refresh}$ is the power consumption of a refresh operation. To find that number, we can use $P_{refresh} = IDD5B * V_{DD}$, where IDD5B is the current consumption of a refresh operation. Here we are assuming that IDD5B is only used for refreshing one row, which is actually not true in real DRAM. Both $IDD5B$ and V_{DD} can be found in the datasheet. There are many different IDD5B parameters to pick from in the datasheet depending on the types of DRAM we are assuming. Assume DDR3-1333 with IDD5B = 160mA, $P_{refresh} = 160mA * 1.5V = 240mW$. Thus, $P_{total} = 0.24W * 2^{47} = 3.3776997^{13}W$. An approximate number is fine.

(d) $Energy = Power * Time$. $T = 24hrs * 60mins * 60secs$, thus $E = 2.9183326^{18}$.

7 Performance Metrics [10 points]

- No, the lower frequency processor might have much higher IPC (instructions per cycle).

  More detail: A processor with a lower frequency might be able to execute multiple instructions per cycle while a processor with a higher frequency might only execute one instruction per cycle.

- No, because the former processor may execute many more instructions.

  More detail: The total number of instructions required to execute the full program could be different on different processors.

8 Performance Evaluation [9 points]

- ISA A: $10\frac{instructions}{cycle} * 500,000,000\frac{cycle}{second} = 5000$ MIPS
- ISA B: $2\frac{instructions}{cycle} * 600,000,000\frac{cycle}{second} = 1200$ MIPS
- Don't know.
  The best compiled code for each processor may have a different number of instructions.