CS 351 Exam 1 Mon. 2/27/2017

Name: _____

Rules and Hints

- You may use one handwritten 8.5 × 11″ cheat sheet (front and back). This is the only additional resource you may consult during this exam. *No calculators.*
- You may write your answers in the form *[mathematical expression][units]*. There is no need to actually do the arithmetic.

Grade

	Your Score	**Max Score**
Problem 1: Short answer		22
Problem 2: Processor performance and power		28
Problem 3: Assembly arrays		25
Problem 4: Assembly functions		25
Total		**100**

Problem 1: Short answer (22 points)

Part A (2 points + 1 if entire class gets it right)

How many bits are in a byte?

Part B (8 points)

Does LEGv8 have fixed-width or variable-width instructions? Give one advantage and one disadvantage of its approach.

Part C (12 points)

Translate these three LEGv8 assembly instructions to 32-bit binary machine code:

```
L:    LDUR X9, [SP, 8]
      ADD  SP, XZR, X19
      CBNZ X9, L
```

Problem 2: Processor performance and power (28 points)

You have a workload with 1 trillion dynamic instructions, with an average CPI of 1.5. You want to run it on a processor with the following characteristics:

- Frequency of 2.2 GHz
- 2 cores
- Static power consumption of 20 W and dynamic power consumption of 40 W

Make the overly simplistic assumptions that your program is the only one running, and you don't have the ability to cut the power to inactive cores. Your program is currently serial (runs on a single core).

Part A: Execution time (8 points)

What is the execution time of your workload on this processor?

Part B: Optimization (10 points)

You have three options for optimizing your program. Give the speedup of each option over the original configuration from Part A. Analyze each option independently, although some of them could clearly be combined.

- Buy a more expensive version of the same processor, which has a clock speed of 2.5 GHz and a voltage that is 10% higher than your original processor's.
- Turn on compiler optimization flags that use more efficient instructions, reducing your CPI to 1.2.
- Parallelize a routine in your program that takes a total of 20% of the execution time, allowing it to run on multiple cores.

Part C: Power and Energy (10 points)

How much power and energy will your processor use in each of the configurations from Part B? Be sure to give units in your answers.

Problem 3: Assembly conditionals and arrays (25 points)

Write the following LEGv8 assembly. At the end of your snippets of code, all registers and memory locations should have their correct values according to the C or English semantics, except for temporary registers.

Part A (10 points)

Assume that the variable a is in X19, b is in X20, and c is in X21.
If $b == c$, then you should make a 1 if the two least significant bits are identical, and 0 otherwise. Otherwise, you should preserve the last (rightmost) 8 bits of a, setting the rest to 0.

Part B (15 points)

Assume that:

- Register X19 contains the base address of an array of 64-bit unsigned long longs.
- Register X20 contains the number of elements in the array.

Write a snippet of LEGv8 code that finds the minimum element in the array and puts it in X21.

Problem 4: Assembly functions (25 points)

Translate the following C functions to LEGv8, using only the instructions on your reference sheet. You should translate each function independently and not attempt to combine them. Obey all LEGv8 conventions about functions, registers, and stack usage.

```
unsigned long long f(unsigned long long a) {
    return a * 4;
}

unsigned long long g(unsigned long long x, unsigned long long y) {
    if (x > y) return f(x) + f(y);
    return f(x + y);
}
```

LEGv8 Arithmetic Instructions

Instruction	Operation	Fmt	Opcode
ADD Rd, Rn, Rm	reg[Rd] = reg[Rn] + reg[Rm]	R	0x458
SUB Rd, Rn, Rm	reg[Rd] = reg[Rn] - reg[Rm]	R	0x658
ADDI Rd, Rn, imm	reg[Rd] = reg[Rn] + imm	I	0x488-0x489
SUBI Rd, Rn, imm	reg[Rd] = reg[Rn] - imm	I	0x688-0x689
ADDS Rd, Rn, Rm	reg[Rd] = reg[Rn] + reg[Rm]	R	0x558
SUBS Rd, Rn, Rm	reg[Rd] = reg[Rn] - reg[Rm]	R	0x758
ADDIS Rd, Rn, imm	reg[Rd] = reg[Rn] + imm	I	0x790-0x791
SUBIS Rd, Rn, imm	reg[Rd] = reg[Rn] - imm	I	0x788-0x789

The versions ending in S also set the Negative, Zero, Overflow, and Carry bits of the FLAGS register.

LEGv8 Logical Instructions

Instruction	Operation	Fmt	Opcode
AND Rd, Rn, Rm	reg[Rd] = reg[Rn] & reg[Rm]	R	0x450
ORR Rd, Rn, Rm	reg[Rd] = reg[Rn] | reg[Rm]	R	0x550
EOR Rd, Rn, Rm	reg[Rd] = reg[Rn] ^ reg[Rm]	R	0x650
ANDI Rd, Rn, imm	reg[Rd] = reg[Rn] & imm	I	0x490-0x491
ORRI Rd, Rn, imm	reg[Rd] = reg[Rn] | imm	I	0x590-0x591
EORI Rd, Rn, imm	reg[Rd] = reg[Rn] ^ imm	I	0x690-0x691
LSL Rd, Rn, shamt	reg[Rd] = reg[Rn] << shamt	R	0x69B
LSR Rd, Rn, shamt	reg[Rd] = reg[Rn] >> shamt	R	0x69A

LSL and LSR replace the shifted-out bits with 0s.

LEGv8 Branch Instructions

Instruction	Operation	Fmt	Opcode
CBZ Rt, CondBrAddr	If (reg[Rt] == 0) PC = BrPC	CB	0x5A0-0x5A7
CBNZ Rt, CondBrAddr	If (reg[Rt] != 0) PC = BrPC	CB	0x5A8-0x5AF
B.cond CondBrAddr	If (FLAGS = cond) PC = BrPC	CB	0x2A0-0x2A7
B BrAddr	PC = BrPC	B	0x0A0-0x0BF
BR Rd	PC = reg[Rd]	R	0x6B0
BL BrAddr	reg[X30] = PC + 4; PC = BrPC	B	0x4A0-0X4BF

BrPC = PC + SignExt ([Cond]BrAddr << 2)

Flags: Negative (N), Zero (Z), Overflow (V), Carry (C)

Category	B.cond	Condition (if SUBS or SUBIS)	B.cond	Condition
Equality	B.EQ	Z = 1	B.NE	Z = 0
Signed < and <=	B.LT	N != V (signed)	B.LE	~(Z = 0 & N = V)
Signed > and >=	B.GT	Z = 0 & N = V	B.GE	N = V
Unsigned < and <=	B.LO	C = 0	B.LS	~(Z = 0 & C = 1)
Unsigned > and >=	B.HI	Z = 0 & C = 1	B.HS	C = 1

LEGv8 Data Transfer Instructions

Instruction	Operation	Fmt	Opcode
LDUR Rt, [Rn, DTAddr]	reg[Rt] = Mem[Rn + SignExt(DTAddr)]	D	0x7C2
STUR Rt, [Rn, DTAddr]	Mem[Rn + SignExt(DTAddr)] = reg[Rt]	D	0x7C0
LDURB Rt, [Rn, DTAddr]	Loads 8b from memory into least significant bits of register	D	0x1C2
STURB Rt, [Rn, DTAddr]	Stores 8b to memory	D	0x1C0

Instruction Formats

R-format:

11b: opcode	5b: Rm	6b: shamt	5b: Rn	5b: Rd

I-format:

10b: opcode	12b: immediate	5b: Rn	5b: Rd

D-format (note: op field is 2b):

11b: opcode	9b: data trans. addr	op	5b: Rn	5b: Rt

B-format:

6b: opcode	26b: branch address

CB-format:

8b: opcode	19b: conditional branch address	5b: Rt

Register List

Name	Use	Needs to be preserved across function call?
X0-X7	Function arguments / results	N
X8	Indirect result location	N
X9-X18	Temporary values	N
X19-X27	Saved values	Y
X28 (SP)	Stack pointer	Y
X29 (FP)	Frame pointer	Y
X30 (LR)	Return address	Y
XZR (31)	Constant value 0	n/a (const.)