CSSE 132 – Introduction to Computer Systems
Rose-Hulman Institute of Technology

Sample Exam 1

KEY

This exam is **closed resources**. You are allowed to use one single-sided 8 1/2 by 11 inch sheet of hand-written notes and a calculator. You may not use a computer, smart phone, etc. during the examination.

Write all answers on these pages. Be sure to **show all work**.

All numbers are expressed in decimal unless specifically stated otherwise.

Write your name and circle your section number on this page, then write your initials on all remaining pages of this exam. You are encouraged to read the entire exam before you start.

You have to turn in the paper part together with your note sheet before you can start the coding part.

	Points available	Your marks
1	9	
2	15	
3	6	
4	5	
5	9	
6	10	
Coding	46	
Total	100	

Problem 1 (9 pts) Convert each of the following **unsigned** binary numbers to *both* hexadecimal *and* decimal. For full credit, be sure to show your work.

(a) `0000 0000 0001 0001`

$$\begin{aligned} Hexidecimal &= 0x0011 \\ Decimal: &= 2^4 + 2^0 = 16 + 1 = 17 \end{aligned}$$

(b) `0000 1000 1000 1000`

$$\begin{aligned} Hexidecimal &= 0x0888 \\ Decimal &= 2^{11} + 2^7 + 2^3 \\ &= 2048 + 128 + 8 \\ &= 2184 \end{aligned}$$

(c) `1000 0000 0000 1111`

$$\begin{aligned} Hexidecimal &= 0x800F \\ Decimal &= 2^{15} + 2^3 + 2^2 + 2^1 + 2^0 \\ &= 2^5 \times 1024 + 8 + 4 + 2 + 1 \\ &= 32768 + 8 + 4 + 2 + 1 \\ &= 32783 \\ or & \\ &= 0x8 \times 2^{12} + 0xF \\ &= 8 \times 4096 + 15 \\ &= 32768 + 15 \\ &= 32783 \end{aligned}$$

Problem 2 (15 pts) For each of the following questions, circle the ONE BEST answer.

(a) Consider the 32-bit int `0xAB917310` stored in a little-endian system. Which byte is stored in the *lowest* memory address?

A. `AB`
B. `91`
C. `73`
D. →`10`

(b) In ARM assembly, which of the following steps is **not** involved with procedure calls (i.e. it is not done by the caller or callee)?

A. The arguments can be passed in `r0, r1, r2, r3`
B. →The stack pointer is set to 0
C. The return address is stored for later use in `lr`
D. Local variables or registers that are needed later are stored on the stack

(c) What tool do we most often use to convert high-level code into assembly code?

A. A text editor
B. A converter
C. An assembler
D. →A compiler
E. None of the above

(d) In C, suppose a string is defined as `char* word = "elephant";`. Which of the following is **false**?

A. `*word` has type `char`
B. The byte in memory after the `'t'` character is a null character `'\0'`
C. →The variable `word` takes up at least 9 bytes of memory
D. `char ch = word[3]` will set char `ch` to have value `'p'`

(e) Given an array `A` of six integers, which statement will increment (add one to) the third element?

A. `*(A+2) = (A+2) + 1;`
B. →`A[2] = 1 + *(A+2)`
C. `A[3]++;`
D. `*(A+(2*4)) = A[2] + 1;`

Problem 3 (6 pts) Answer the following questions about addressing memory. Be sure to show your work for full credit when you need to compute answers.

(a) What is the smallest addressable unit of data in modern computer systems?
8 bits, or one byte

(b) If your computer uses 16-bit addresses, how many unique addresses can it access?
2^{16} *Bytes, or* 2^6 *Kilobytes, or* 64 *Kilobytes.*

(c) How many address bits do you need to address 100,000 unique things?
$2^{16} = 65536$ *and* $2^{17} = 131072$*, so you need 17 bits*

Problem 4 (5 pts) A friend of yours gives you a free hard disk with the following specifications:

Capacity:	30 Terabytes
Average Access Time:	10ms

You've purchased a cache for this hard disk. The cache takes 100 ns to access. Your computer puts often–accessed data from the hard disk into the cache and 97.88% of time, the desired data is ready in the cache. If the data is not found in the cache it must be read off the disk.

If you always check the cache before attempting to load the data from disk, what is the new average access time for data using this hard drive with the cache? Show your work for full credit.

$$\begin{aligned} avgtime &= 0.9788 \times time_{cache} + (1 - 0.9788) \times (time_{cache} + time_{disk}) \\ or &= 1.0 \times time_{cache} + 0.0212 \times time_{disk} \\ &= 100\text{ns} + 0.0212 \times 10.0\text{ms} \\ &= 100\text{ns} + 0.0212 \times 10,000,000\text{ns} \\ &= 100\text{ns} + 212000\text{ns} \\ &= 212100\text{ns} \end{aligned}$$

Problem 5 (9 pts) The following table shows information about multiple types of memory.

	Avg. Access Time	Price
SRAM	< 1 ns	\$200/GB
DRAM	10 ns	\$5.00/GB
Flash Memory	0.5 ms	\$0.50/GB
Hard Disk	10 ms	\$0.10/GB
Cloud Storage	100 ms–2 s	\$0.01/10 GB

(a) Modern computer processors issue billions of small read and write operations per second. Which type of memory would you choose to build a register file in your CPU? Justify your answer.

SRAM since the writes are small, but require very fast access.

(b) Uncompressed video often requires 50GB-100GB of total storage, but is usually accessed in smaller, easy to manage blocks. Which type of memory would you choose as the primary storage for such video (where most of it will stay)? Justify your answer.

Flash or hard disk since the total size is large, but the data can be accessed in convenient chunks.

(c) Interactive programs like web browsers and video games require often fast access to large amounts of memory (2GB-5GB). What type of memory would be best for interactive applications? Justify your answer.

DRAM, since the size is large, but fast access is still required.

Problem 6 (10 points) Read this ARM assembly code and answer the following questions. A simplified ARM guide is provided on the last page for your reference.

```
1  mystery
2    mov r1, #0      ; sum=0:
3    mov r3, #0      ; val=0
4    b .L2           ;
5  .L3:
6    add r1, r1, r3 ; sum += val
7    add r3, r3, #1 ; val += 1
8  .L2:
9    cmp r3, #9      ; while val <= 9
10   ble .L3
11   mov r0, r1     ; return value goes into r0
12   b   lr
```

(a) Given the following initial register values below, fill in the values in all the registers after the above code finishes executing.

Register	initial value	final value
r0	0	45
r1	0	45
r2	0	0
r3	0	10

(b) Write a C function for `mystery` that does something equivalent to this assembly code. Assume `mystery` takes no parameters, and also assume local variables are not stored on the stack (registers are used instead).

```
1  int mystery() {
2    int sum = 0;
3    int val = 0;
4    while (val <= 9) {
5      sum = sum + val;
6      val = val + 1;
7    }
8    return sum;
9  }
```

Basic ARM Assembly Guide

Register	Purpose
`lr`	holds return address
`sp`	address of stack top
`r0-r10`	general purpose registers
`r0`	also used for return value
`r0-r4`	also used for arguments

Instruction	What it does
`str x, Mem`	Store x's value into memory at address Mem...
`str x, [y]`	... where Mem is an address (pointer) stored in register y
`str x, [y, #a]`	... where Mem is a + the address (pointer) stored in register y
`ldr x, Mem`	Load memory at address Mem into x
`ldr x, [y]`	... where Mem is an address (pointer) stored in register y
`ldr x, [y, #a]`	... where Mem is a + the address (pointer) stored in register y
`mov x, y`	Copy y's value into x
`sub x, y, z`	x = y − z
`add x, y, z`	x = y + z
`cmp x, y`	compare x to y, set conditions
`b LABEL`	go to LABEL (unconditionally)
`bl LABEL`	call procedure at LABEL
`bx lr`	return
`blt LABEL`	go to LABEL if condition says x <y
`ble LABEL`	go to LABEL if condition says x ≤ y
`bgt LABEL`	go to LABEL if condition says x > y
`bge LABEL`	go to LABEL if condition says x ≥ y