Computer Architecture I Midterm I

Chinese Name: ________________________________

Pinyin Name: ________________________________

E-Mail ... @shanghaitech.edu.cn: ________________________________

Question	Points	Score
1	1	
2	12	
3	16	
4	14	
5	18	
6	17	
7	22	
Total:	100	

- This test contains 8 numbered pages, including the cover page, printed on both sides of the sheet!.
- We will use gradescope for grading, so only answers filled in at the obvious places will be used.
- Use the provided blank paper for calculations and then copy your answer here.
- Please turn **off** all cell phones, smartwatches, and other mobile devices. Remove all hats and headphones. Put everything in your backpack. Place your backpacks, laptops and jackets under your seat.

- You have 85 minutes to complete this exam. The exam is closed book; no computers, phones, or calculators are allowed. You may use one A4 page (front and back) of notes in addition to the provided green sheet.
- The estimated time needed for each of the 6 topics is given in parenthesis. The total estimated time is 80 minutes.
- There may be partial credit for incomplete answers; write as much of the solution as you can. We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided.
- Do **NOT** start reading the questions/ open the exam until we tell you so!
- Unless otherwise stated, always assume a 32 bit machine for this midterm.

$\boxed{1}$ 1. First Task (worth one point): Fill in you name
Fill in your name and email on the front page and your ShanghaiTech email on top of every page (without @shanghaitech.edu.cn) (so write your email in total 8 times).

2. Various Questions (12 pts; 10 min)

3 (a) Name 6 Great Ideas in Computer Architecture.

2 (b) You define a short recursive MIPS procedure `foo` that is statically linked by two executables. Can the binary for the procedure `foo` be different in the two executables? Why, or why not?

2 (c) What is the difference between the `add` and `addu` MIPS instructions?

(c) ______________________________

2 (d) Which MIPS registers are preserved over a function call (write all register names - e.g. $m2-$m6)?

(d) ______________________________

1 (e) How many things can you represent with N bits?

(e) ______________________________

2 (f) In the lecture you learned about CALL (this is the CALL regarding executing a C program on a computer). Name (each a single word) the important steps of this procedure in the right order.

3. Number Representation (16 pts; 15 min)
Please convert the following 8-bit Signed Integers into decimal form.
Explicitly write '+' and '-'.
For example, suppose that the binary representation is 01000001B and it is represented in Sign-Magnitude Representation, then the solution is +65D.

$\boxed{2}$ (a) If represented in Sign-Magnitude Representation

`Suppose the binary representation is 10000001B:` ______________

`Suppose the binary representation is 10000000B:` ______________

$\boxed{2}$ (b) If represented in 2's Complement Representation

`Suppose the binary representation is 10000001B:` ______________

`Suppose the binary representation is 11111111B:` ______________

$\boxed{4}$ (c) Suppose a is an $8-$bit signed integer represented as a_{hex} = 0xCB, then its binary representation is a_{two} = ____________________, its decimal representation is a_{ten} = ________.

$\boxed{4}$ (d) For an 10-bit value, two's complement integer, what are the largest AND smallest value you can represent in decimal?

$\boxed{4}$ (e) Assume an 8-bit two's complement machine on which all operators are performed on 8-bit registers. Answer the results of the following operations in hexadecimal. Assume that subtraction is done with SUBU and addition is done with ADDU.

a
$$\begin{array}{rl} & \text{8C (hex)} \\ - & \text{B5 (hex)} \\ \hline & \end{array}$$

b
$$\begin{array}{rl} & \text{8A (hex)} \\ + & \text{3E (hex)} \\ \hline & \end{array}$$

4. C programming I (14 pts; 14 min)

4 (a) What is the value of **s** in the following code? If possible, also provide the actual number in decimal.

```
unsigned int t[] = {0, 1, 2, 3, 4, 5};
unsigned int s = sizeof(t);
```

(I) The length of the **t** array.
(II) The number of bytes in the t array.
(III) The number of bytes in one unsigned int.
(IV) The number of bytes in an unsigned int pointer.
(V) Nothing: an error will be produced.

(a) ______________________________

5 (b) What does the following method do?

```
char * func(char *f, char y){
  char *h = f;
  for(h = f; *h != y && *h){
    h++;
    if(*h){
      *h = 0;
      h++;
    }
  }
  return h;
}
```

(I) It returns a pointer to the first location of **y** in **f**.
(II) It splits **f** at the first occurrence of **y** and then returns a pointer for the remaining string.
(III) It finds the last location of **y** in f, zeros put that location, and then returns a pointer to the next location.
(IV) It zeros out **f** until it finds **y**. It then returns a pointer to the location of **y** in **f**.
(V) Nothing; it cannot be complied.

(b) ______________________________

5 (c) The **%s** format specifier takes in a **char*** and prints until it finds a null character. What do the following two lines of code print?

```
char *s = "uncharacteristic";
printf("%s", s+s[7]-s[6]);
```

(c) ______________________________

5. C programming II (18 pts; 10 min)

9 (a) Given the code below:

```
int a;
const char * b;
int foo(short c){
   char d;
   static int * e = malloc(sizeof(int));
}
int main(){
   int f;
   foo(3);
}
```

Name all areas of memory that are used by the program:

(a) ____________________

At which area of the memory are the following variables stored:

a:	b:
c:	d:
e:	f:

9 (b) 1. When passing parameters to functions, what is the difference between **call by value** and **call by reference**?

2. What is the default function call method in (I) MIPS and (II) C?

3. How can we achieve the other call method?

6. Bits and Pieces (17 pts; 15min)

   Read the following MIPS assembly code and answer the following questions. Your answers should be as concise as possible.

```
halo:
 #BEGIN
 addiu $v0, $0, 0
 addiu $s0, $0, 0
 addiu $s1, $0, 0
 beq $a0, $0, finish_the_fight
 addiu $s0, $a0, 0
 addiu $s1, $0, 1
 andi $t0, $s0, 1
new_mombasa:
 bne $t0, $0, finish_the_fight
 srl $s0, $s0, 1
 sll $s1, $s1, 1
 andi $t0, $s0, 1
 j new_mombasa
finish_the_fight:
 addiu $v0, $s1, 0
 #END
 jr $ra
```

4 (a) Briefly explain what `halo` returns with respect to the input.

7 (b) Try to implement the function in C as efficient and concise as possible. (The space given below is more than enough.)

6 (c) We've broken some assembly language calling conventions with the code above. Write the code that should be inserted at the positions of *#BEGIN* and *#END* to correct it.

At **#BEGIN**:

At **#END**:

7. MIPS & Branch-if-equal instruction (22pts; 16 min)

10 (a) True/False: circle the correct answer (2 pts each)

T	**F**	**1.**	Branch instructions in MIPS can only jump forward 32768 and backward 32767 instructions.
T	**F**	**2.**	A carry-out at the most significant bit after an addition of two signed numbers always indicates overflow.
T	**F**	**3.**	I-type instructions always cause pipeline bubbles.
T	**F**	**4.**	If we only have three parameters to send to a non-recursive function, then we can use registers and don't need to use the stack.
T	**F**	**5.**	Every location in the text segment is accessible from a single branch statement.

$\boxed{6}$ (b) Consider a hypothetical branch-if-equal instruction that is 32 bits long:
- 6 bits are used to encode the opcode
- 6 bits are used to encode one register number
- 6 bits are used to encode another register number
- 14 bits are used to encode an offset that will be added to the program counter (PC) if the branch ends up being taken, and a new instruction address is required. (The number is not in 2's complement form, and all 14 bits can encode a constant.)

Thus, the instruction syntax might be: `BEQ R12, R11, X`
- If R12==R11, the PC will be set to PC+X instead of PC+4.

Given this instruction, is the code shown in the table below valid? Why or why not? Explain in detail.

Address	**Instruction**
5000	...
5004	BEQ R12, R11, X
5008	Add R1, R2, R3
...	...
X: 21256	Sub R1, R2, R3

$\boxed{6}$ (c) Instruction Format: Translate the assembly into machine code and vice versa (use named registers - not register numbers).

Instruction	Code (hex)
lb $t3, 7($s5)	
	0x00128a00