Computer Architecture I Mid-Term I

Chinese Name: _____

Pinyin Name: _____

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

Question	Points	Score
1	1	
2	11	
3	10	
4	9	
5	14	
6	29	
7	14	
8	12	
Total:	100	

- This test contains 25 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 out of reach.
- You have 120 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 handwritten notes in addition to the provided green sheet (one of those can be printed).
- The total estimated time is 120 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 exam.

[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 25 times).

2. **Various Questions**

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

Solution: 1. Abstraction (Layers of Representation/Interpretation)
2. Moores Law (Designing through trends)
3. Principle of Locality (Memory Hierarchy)
4. Parallelism
5. Performance Measurement and Improvement
6. Dependability via Redundancy

[1] (b) Which registers will not be preserved over a function call? (Only Considering $\$t$, $\$a$, $\$s$, $\$v$, $\$at$ and $\$sp$, same range of registers for the two following problems.)

Solution: $\$t$, $\$a$, $\$v$, $\$at$

[1] (c) Which registers will be preserved in the stack while we do function call?

Solution: $\$s$

[6] (d) Let's play with CALL!
Connect the definition with the name of the process that describes it. (Please fill in the blanks before 1) to 9))
a) Compiler b) Assembler c) Linker d) Loader

__ 1) Outputs code that may still contain pseudoinstructions.

__ 2) Takes binaries stored on disk and places them in memory to run.

__ 3) Makes two passes over the code to solve the "forward reference" problem.

__ 4) Creates a symbol table.

__ 5) Combines multiple text and data segments.

__ 6) Generates assembly language code.

__ 7) Generates machine language code.

__ 8) Only allows generation of TAL.

__ 9) Only allows generation of binary machine code.

__ 10) Resolves relative addressing.

__ 11) Resolves absolute addressing

__ 12) Which may make use of at register?

Solution: 1) a
2) d
3) b
4) b
5) c
6) a
7) b
8) b
9) c
10) b
11) c
12) b

3. **Number Representation**
Fill in the blanks below

4 (a) Given a 10-bit binary number, what is the range of the integer it can represent?

If unsigned smallest: _____ largest: _____

If one's completement smallest: _____ largest: _____

If two's completement smallest: _____ largest: _____

Solution: 0 1023
-511 511
-512 511

4 (b) Convert 2018_{ten}

To Binary _____

To Hexadecimal _____
To a base-15 number [A(10), B(11), C(12), D(13), E(14)]

_____

Solution: 11111100010 $7e2$ $8e8$

2 (c) Concisely describe how to identify **overflow**

Solution: Carry into MSB $\neq$ to Carry out MSB

4. **Memory With C**

Note: The following code is complied with "-m32 -std=c89"

2 (a) Suppose we have defined the C structure:

```
struct student {
  int id;
  int score;
  char name[8];
};
```

Also, we declare:

```
struct student students[3];
struct student *studentTwo = students + 2;
```

Suppose that students starts at $0x10000000$. What is the value of studentTwo?

(a) **0x10000020 (0x20**$= 32_{ten}$**)**

We are creating songs in preparation of the graduation party. Consider the following program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Song {
  char *title;
  char *artist;
} Song;

Song * createSong() {
  Song* song = (Song*) malloc(sizeof(Song));
  song->title = "this old dog";
  char artist[100] = "mac demarco";
  song->artist = artist;
  return song;
}

int main(int argc, char **argv) {
  Song *song1 = createSong();
  printf("%s\n", "Song written:");
  printf("%s\n", song1->title); // print statement #1
  printf("%s\n", song1->artist); // print statement #2
  Song song2;
  song2.title = malloc(sizeof(char)*100);
  strcpy(song2.title, song1->title);
  song2.artist = "MAC DEMARCO";
```

```
27   printf("%s\n", "Song written:");
28   printf("%s\n", song2.title); // print statement #3
29   printf("%s\n", song2.artist); // print statement #4
30   return 0;
31 }
```

5 (b) In the following code is listed that prints a certain address. In which part of the memory will this address be located? (Circle it)

1. `printf("%p",song1);`
   (a) Stack
   (b) Heap
   (c) Static
   (d) Code
2. `printf("%p",song1->title);`
   (a) Stack
   (b) Heap
   (c) Static
   (d) Code
3. `printf("%p",song1->artist);`
   (a) Stack
   (b) Heap
   (c) Static
   (d) Code
4. `printf("%p",&song2);`
   (a) Stack
   (b) Heap
   (c) Static
   (d) Code
5. `printf("%p",song2.title);`
   (a) Stack
   (b) Heap
   (c) Static
   (d) Code

Solution: b, c, a, a, b

2 (c) Will all print statements execute as expected? Circle: YES NO

Solution: NO

If you answered yes, leave this blank. If you answered no, write the number(s) of the print statement(s) which will not execute as expected.

(c) **2**

5. **C Programming (Variables & pointers)**

All questions in this section are based on the following code.
Assume the code is natively compiled on a 64-bit system.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <stdint.h>
4
5   long int exam_foo(const uint32_t * array_foo){
6       return sizeof(array_foo);
7   }
8
9   int main(){
10      uint32_t a = 0;
11      uint32_t * b = &a;
12      uint32_t ** c = &b;
13
14      uint32_t arr[5] = {0x2018,0x4,0x1,0x110,0x2019};
15      uint32_t * p = arr;
16  }
```

7 (a) What's the size of the following variables? (Your answer should be in bytes.)

a:

b:

c:

arr:

arr[0]:

p:

array_foo:

Solution:

1. 4
2. 8
3. 8

4. 20
5. 4
6. 8
7. 8

2 (b) What type does **sizeof(array_foo))** return? What is the value of **sizeof(sizeof(array_foo))**?

(b) **size_t; 8**

5 (c) What's the expected value of the following expression. If an expression might cause an error, write "ERROR". For all numbers, write them down in their hexadecimal form. (0x)

*(p+3):

p[2]:

*(p+2) + p[1]:

*(uint16_t *)p[0]:

(uint8_t)(p+4):

Solution:

1. 0x110
2. 0x1
3. 0x5
4. ERROR
5. 0x19

6. **MIPS**

Solution: Est. Avg. score for MIPS part: 19/27 approx. 70.37%

Notice. In this part (especially parts (d) and (e)), you can write at most **ONE** line of code in each space when we ask you to write down codes, but you do not have to use all of the spaces. **If you write more than one line of code in one space, that answer will be voided.** ("one line of code" means one semicolon in C or one instruction in MIPS)

5 (a) This subquestion involves T / F questions. Circle the correct answer.

T / F: li is a pesudo instruction.

T / F: There are at most 15 tailing bits with value zero in $\$t0$ after the execution of the following instruction:

```
1       lui   $t0,  $t0,  0x8000
```

T / F: The value in $\$t0$ and $\$t1$ are the same after the execution of the following instruction:

```
1       addi  $a0,  $zero, 10
2       sra   $t0,  $a0,  1
3       srl   $t1,  $a0,  1
```

T / F: You can jump to any instruction you like using a J type instruction as long as the label is provided.

T / F: Instruction *addi* can cause overflow exception.

Solution: TFTFT
2) False, at most 31 tailing zeros. This comes from lab 3.
4) False, you can't. there are only 26(+2) bits to write an address.

Worth 1 pt each.
Est. Avg.: 4.5

6 (b) Please translate the following instruction from MIPS to hex value and vice versa. Besides, specify which type of instructions are these.

```
1       addu  $t0,  $t9,  $s1
2       jr    $ra
3       0x8fb00020
4       0x014b4826
```

Translation of Line 1: _____, instruction type: _____

Translation of Line 2: _____, instruction type: _____

Translation of Line 3: _____, instruction type: _____

Translation of Line 4: _____, instruction type: _____

Solution: Translation should account for 1 credit each, but the type should account for 0.5 each, or 1 credit should they get it all correct, as they are easy. Only Hex value is accepted, as it is stated in the instruction above.

```
1       0x03314012              # R type
2       0x03e00008              # J type
3       lw    $s0,  32($sp)     # I type
4       xor   $t1,  $t2,  $t3   # R type
```

Each translation worth 1 pt, the first 3 instruction type worth 1 in total(deduct 0.5 for each wrong), the last instruction type worth 1 pt.
Only Hex value is accepted, as it is stated in the instruction above.
Est. Avg.: 4

3 (c) Creative Instructions.
Write a MIPS instruction which will cause an infinite loop if ever executed, anywhere, in any MIPS program.
(Both instruction and Hex value along with it's type)

Instruction: _____

Hex value: _____

Solution: The label can be set to anything, not only "Inf" is allowed. The instruction worth 1 pt and hex value worth 2 pt.
Easy way to tell if his hex value is correct: it must ends with 0xffff(-1), or it will not cause inf loop.
Instructions below are acceptable.

```
1       Inf0: beq         $zero,  $zero, Inf0 # 0x1000ffff
```

```
Inf1: bne          $fp,  $sp,  Inf1  # 0x17ddffff
```

Origination: https://tbp.berkeley.edu/exams/4581/download/ Q2(d)
Est. Avg.: 1

7 (d) Let's talk about branches.
1.) You may have used *ble* in your projects. But it is known as a pseudo instruction. For example the following instruction. Please write down the complete RTL for it.

```
ble   $t0,  $t1,  Label
```

Solution: If *$t0* $\leq$ *$t1*, jump to *Label*

```
if(R[rs]<=R[rt]) PC = Label
  else PC = PC + 4
```

Worth 1 pt.
Est. Avg.: 1

2.) Pseudo instructions can not be compiled into machine code directly, it has to be translated to some real instructions beforehand. Please come up with **two TAL instructions** that can be used to **translate the pseudo instruction above**.
The instructions you write should be able to **substitude** any *ble* pesudo instruction anywhere in a MIPS program.
Hint: If you can't think of a way to explain that pseudo instruction in 2 instructions, you can write a solution with 3 instructions. We will deduct some pts but not all of them.

```
      _____________________________________

      _____________________________________

      # Think harder before you write anything below.

      _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

Solution:

```
2Inst:
      slt   $at,  t$1,  $t0
      beq   $at,  $zero,  Label
3Inst:
```

```
addiu  $at,  $t1,  1
slt    $at,  $t0,  $at
bnq    $at,  $zero,  Label
```

Worth 2 pt. If 3 instructions are used, deduct 1 pt.(Or 0.5 depends how they are doing.)
Est. Avg.: 1.5

3.) In 2.), why don't we use *$t2* or *$t3* as a temporary register? Which register did we use instead? What is the purpose of using this one?

Solution: 1. There is no guarantee that *$t2* is available anywhere in a MIPS program. 2. *$at* is a temporary register used by assember, it cannot be used by the user so it should be available anywhere. Two questions, must have 2 answers. Worth 2 pts.
Est. Avg.: 1

8 (e) StarCraft is a famous RTS(Real Time Strategy) game. The first edition was issued in 1998 and it has been played by *zealots* all over the world for 20 years. In this game, you will be commanding an army to fight your enemy until he/she has no buildings left (or, unfortunately, you have nothing left).
To build such a game, you need a combat system. In the simplest case, when two units meet, they attack each other until one dies, thus both of them must have health points (hp) and damage. When one side has 0 or negative hp, we consider it dead. Notice the possibility that two units may die at the same time (both of them throw a critical strike to each other at the same time).
Let's assume two units (call them unit0 and unit1) meet, they attack each until one or both die. You should report which one died (0 or 1). If both of them died, report -1.
In this part, you will be filling unfinished C and MIPS code.

```
const bool bothAlive(
   const int hp0, const int hp1){
   return (hp0>0 && hp1>0);
}
const int attack(
  int hp0, int hp1,
  const int damage0, const int damage1){
    while (bothAlive(hp0, hp1)){
       hp0 -= damage1;
```

```
10          hp1 -= damage0;
11      }
12      if (hp0 <= 0 && hp1 <= 0){
13          return -1;
14      } else {
15
16          ____________________________________
17      }
18  }
```

```
1  bothAlive:
2        slt   $t0,  $zero, $a0
3        slt   $t1,  $zero,  $a1
4
5        ____________________________________
6        jr    $ra
7
8  attack:
9        sw    $ra,  0($sp)
10       addi  $sp,  $sp,  -4
11       LoopChecker:
12             jal bothAlive
13
14             ____________________________________
15       Loop: subu  $a0,  $a0,  $a3
16             subu  $a1,  $a1,  $a2
17             j     LoopChecker
18       EndLoop:
19
20       ____________________________________
21
22       ____________________________________
23
24       and   $t3,  $t0,  $t1
25       if:   bne   $t3,  1,    else
26             addiu $v0,  $zero, -1
27             j endIf
28       else:
29
30             ____________________________________
31       endIf:
32       addi  $sp,  $sp,  4
33       lw    $ra,  0($sp)
34
35       ____________________________________
```

a) Please fill in the C code.
b) Please fill in the (half-translated) MIPS code.
c) Your peer thinks that function *bothAlive* is wrong as *$ra* is not saved beforehand. Please write the instructions to make it right or argue with your peer about why he is wrong.

Solution: a)

```
return (hp1 > 0);
```

b)

```
and   $v0,  $t0,  $t1

beq   $v0,  0, EndLoop

sle   $t0,  $a0,  $zero
sle   $t1,  $a1,  $zero

slt   $v0,  $zero, $a1

jr    $ra
```

c) Disagree. There is no function call in *bothAlive* - it is a leaf function - , so there is no need to save *$ra*

All questions worth 1 pt each.
Est. Avg.: 6

7. C Programming

8 (a) This subquestion involves T / F questions. Incorrect answers on T / F questions are penalized with negative credit. Circle the correct answer.

T / F: Every C Program must have the statement `#include <stdio.h>`.

T / F: A memory leak can always be detected because the program crashes whenever one is present.

T / F: The ASCII values for the standard characters go from 1 to 128.

T / F: Static memory means that it exists throughout the execution of a program.

T / F: Given the array `char letters[26]`, `letters` is the address of `letters[0]`

T / F: If `ptr2` is set to `ptr` (a pointer given by the declaration "`char ptr[10]`") then `ptr2++` points to the cell `ptr[1]`

T / F: The following is a legal macro:

```
#define Love printf("I love Computer Architecture!\n")
```

T / F: If `Yang` is a pointer to a structure that has an `int*` variable age with `*age = 21`, then to access the value of `age` we can write:

```
Yang -> (*age);
```

T / F: If we are given `char str[] = "Rua!"` then the command `printf("%s", str+2)` will print out "a!"

T / F: If we are given `char str[] = "Rua!"` then the command `printf("%s", ++str)` will print out "ua!"

Solution: FFFTT TTFTF

6 (b) You are asked to allocate and free a 2 dimensional array dynamically. You do not need to check memory validity when you are using malloc. The 2 dimensional array `a` should be accessed as `a[Row][Col]` (index of row starts from `0`, ends with `nRow-1`; index of column starts from `0`, ends with `nCol-1`).
There is a way to program this such that during execution the **free** function will be called only 2 times, regardless (independent) of the size of nCol and nRow. If you implement this version you can get full points, otherwise you can only get 4 points for this question.

```
int **a;
int i, iRow, iCol;
/* Allocate a 2 dimensional array with nRow rows and nCol
   columns using malloc */

_____________________________________________

_____________________________________________

_____________________________________________

_____________________________________________

_____________________________________________

_____________________________________________

for (iRow = 0; iRow < nRow; iRow++) {
      for (iCol = 0; iCol < nCol; iCol++) {
            a[iRow][iCol] = iRow + iCol;
      }
}

/* Free the 2 dimensional array */

_____________________________________________

_____________________________________________

_____________________________________________

_____________________________________________

_____________________________________________
```

Solution:

```
/*allocate*/
a = (char **) malloc(sizeof(char *) * nRow);
a[0] = (char *) malloc(sizeof(char) * nRow *
   nCol);
for (i = 1; i < nRow; i++) {
      a[i] = a[i-1] + nCol;
}
/*free*/
free(a[0]);
```

```
9               free(a);
```

8. **C Programming**

2 (a) Fill in the declaration of a single linked linked-list below.

```
1  /* definition of single linked list */
2  typedef struct node{
3     int value;
4     /* a pointer to the next node */
5
6
7     ______________________________ next;
8  } node;
```

Solution: struct node *

5 (b) You are asked to convert an int array to a single linked list. All variables you are allowed to use have been defined in the beginning of the function.

```
1  /* turn an array to linked list */
2  /* if the array is empty, just return NULL */
3  node * int_arr_to_list(const int * arr, int len){
4     /* declare all variables you might use */
5     node * header;
6     node * curr;
7     int i;
8
9     /* student should fill in everything here */
10    if (len == 0)  return NULL;
11
12    /* deal with the header */
13
14
15    header = _____________________________________
16
17
18    header->value = _____________________________
19
20
21    header->next = ______________________________
22
23    /* fill in the rest */
24    curr = header;
25
26
27    ____________________________________________
28
29
30    ____________________________________________
31
32
33    ____________________________________________
34
35
36    ____________________________________________
37
38
39    ____________________________________________
40
41    return header;
42 }
```

Solution:

```
/* turn an array to linked list */
/* if the array is empty, just return NULL */
node * int_arr_to_list(const int * arr, int len){
   /* declare all variables you might use */
   node * header;
   node * curr;
   int i;


   /* student should fill in everything here */
   if (len == 0)  return NULL;

   /* deal with the header */
   header = malloc(sizeof(node));
   header->value = arr[0];
   header->next = NULL;

   curr = header;
   for (i = 1; i < len; i++){
      curr->next = malloc(sizeof(node));
      curr->next->value = arr[i];
      curr->next->next = NULL;
      curr = curr->next;
   }

   return header;
}
```

5 (c) Below is code to append a new node after the end of a linked-list. Part of the code is wrong, read the code and answer the following questions.
Note: Suppose main() and other supporting functions are bug-free. Header files have been included properly.

```
1  /* append an element to end of the list */
2  void append_node(node * list_head, int val){
3     node * curr = list_head;
4     node new_node;
5     while(curr->next != NULL){
6        curr = curr->next;
7     }
8
9     new_node.value = val;
10    new_node.next = NULL;
11    curr->next = &new_node;
12 }
```

1. Can the program successfully compile (produce execuable file), if the above function is not called but presented in the source code?

2. Can the above program successfully compile (produce execuable file) if the above function is called?

3. Point out the error and correct it in the following format.
Example: `Line: 3 | Correction: node * new_curr = list_header;`

Solution:

1. Yes, it can compile.
2. Yes, it can still compile.
3. The reference corrected function is below:

```
/* error: return of value on stack */
/* append an element to end of the list */
void append_node_2(node * list_head, int val){
   node * new_node = malloc(sizeof(node));
   node * curr = list_head;
   while(curr->next != NULL){
      curr = curr->next;
   }

   new_node->value = val;
   new_node->next = NULL;
   curr->next = new_node;
}
```