Lab 3 : MIPS Control Instructions

Name:

Your Name

Sign the following statement:

On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work

Type Name

1 Objective

The objective of this lab is to make you comfortable with MIPS control instructions by writing, running, and debugging several complex programs.

2 Pre-requisite

Before starting with this lab, you should be familiar with MIPS control instructions as well as MARS.

3 Control Instructions

As you know, the `PC` points to the current instruction; however, for efficiency reasons, the `PC` is actually updated to the next instruction before the current instruction has completed

its execution. This means that the offset used in computation of PC-Relative addressing is actually relative to the next instruction instead of the current one. The assembler in MARS hides this complexity from the user when it computes offsets in instructions, but you can see this if you examine the instructions it creates.

1. Enter the following code into a file named *lab3-1.s* and run it using MARS. Use it to answer the questions below:

```
1            .text
2            .globl main
3
4            main: li       $a1, 10
5            add $t0, $zero, $zero
6            loop:   beq $a1, $zero, finish
7            add $t0, $t0, $a0
8            sub $a1, $a1, 1
9            j    loop
10           finish: addi $t0, $t0, 100
11           add  $v0, $t0, $zero
```

   (a) Step through the program and find the branch instruction. Locate the value of the 16-bit branch address and fill in Table 1.

Tab. 1: Branch Instruction Address (when delayed branches is disabled)

Instruction	Address field in instruction (16 bits)	Target Address
beq $5,$0,4	0000 0000 0000 0100	0x00400018

   (b) Step further through the program and find the jump instruction. Locate the value of the 26-bit jump address and fill in Table 2.

Tab. 2: Jump Instruction Address (when delayed branches is disabled)

Instruction	Address field in instruction (26 bits)	Address field shifted left by 2 bits (28 bits)	Target Address
j loop	00000100000000000000	0000010000000000000000	0x00400008

2. (a) Write a MIPS program with the following specifications:

   - Reserve space in memory for a variable called UIN of size word. The initial value of UIN will be the sum of the digits in your UIN.
   - The program should implement the following piece of C-code:

```
for(i=0; i<10; i++)
UIN = UIN-1;
```

   - Save your file as "*lab3-2.s*", and make sure you comment the code.

   (b) What is the value of UIN before and after executing the code:

Variable	Value before the run	value after the run
UIN	28	18

   (c) Demonstrate your program to the TA; ☑.

3. Create a program with the following specifications:

   - Reserve space in memory for an array of **words** of size 10. Use the '.space' directive. The array is called **my_array**.
   - The program should implement the following piece of C-code. The value of initial_value is the first digit of your UIN.

```
    int j = initial_value;
    for (i=0; i<10; i++)
    my_array[i] = j;
    j++;
}
```

   - Save your file as "*lab3-3.s*", and make sure you comment the code.
   - Demonstrate your program to the TA; ☑.

   Hint: A common mistake here is to forget that when addressing words in memory, any word will be located 4 bytes after the previous word, not 1.

4 Deliverables

- Submit completed copy of this lab manual.
- Include the following in your eCampus submission:
  - The source code for all .s files.