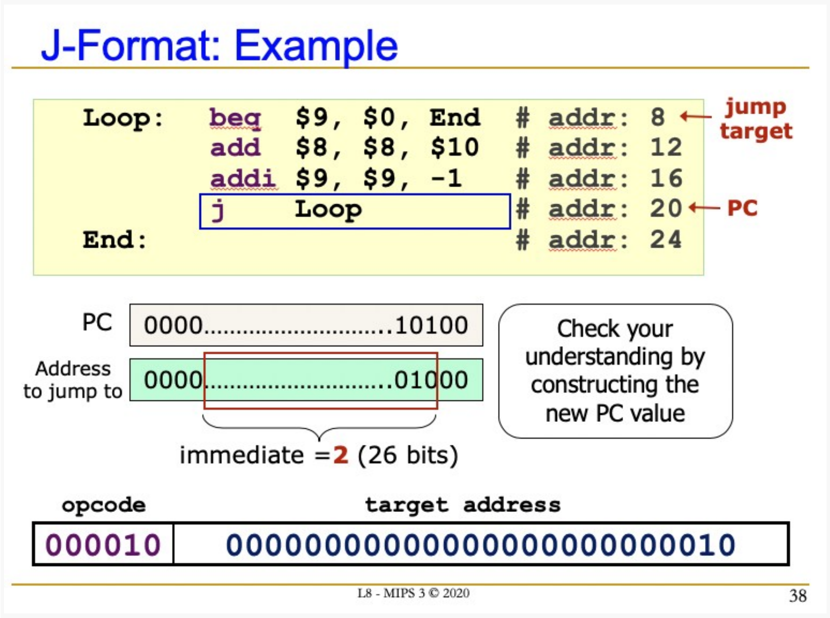
Eric Samuel Huddleston hpg103
Jose Carlos Gomez-Vazquez brs849
Marcus Toudouze gdp629
Section 002
10/20/2020

CS 3853 Computer Architecture Term Project Part 3 Fall 2020

Part 3 (20%)

1. Study the L8-MIPS 3 slide deck on MIPS J instruction format - slide number 33 to 38.

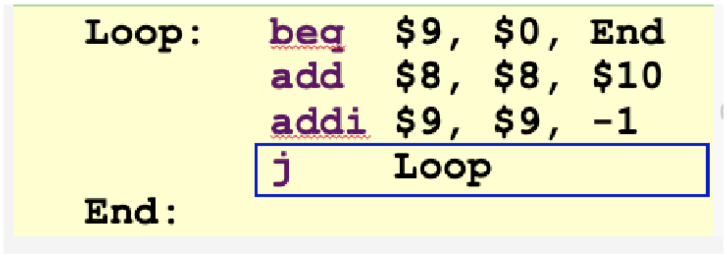
2. Study and understand the J instruction encoding on slide number 38.



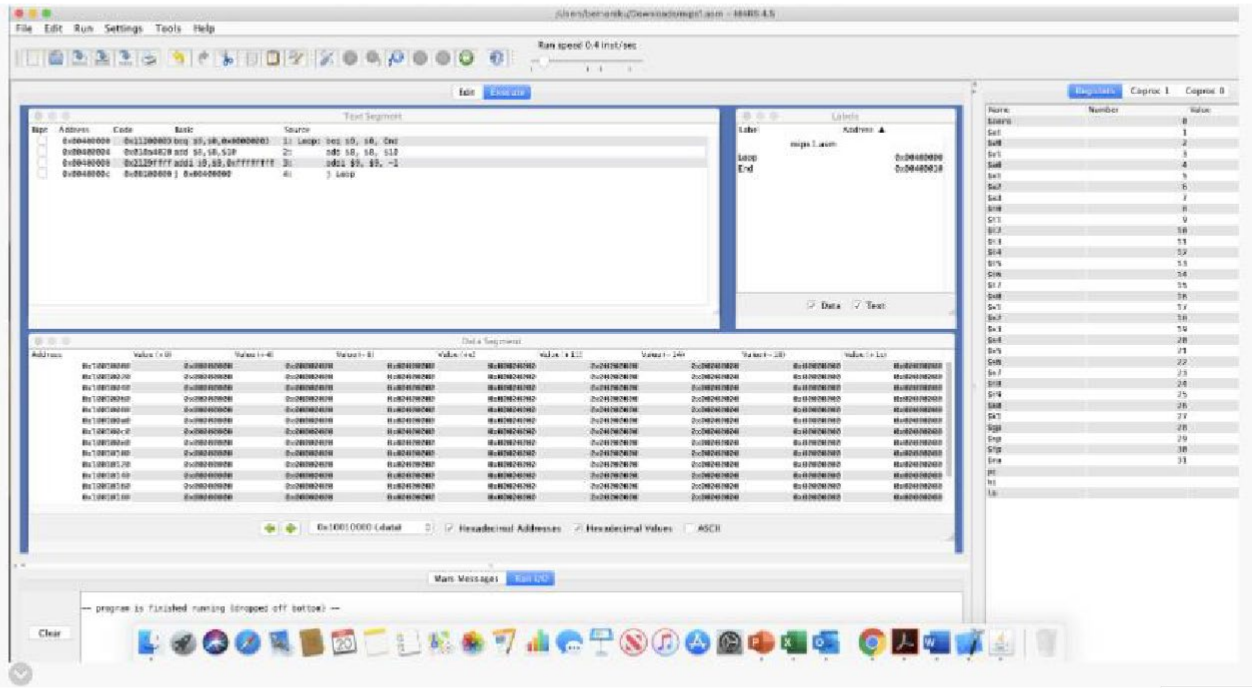
J-Format: Example

```
Loop:  beq   $9, $0, End    # addr: 8  ← jump target
       add   $8, $8, $10    # addr: 12
       addi  $9, $9, -1     # addr: 16
       j     Loop           # addr: 20 ← PC
End:                        # addr: 24
```

3. Edit the following codes and saved in a program called Loop.asm and run it on MARS.



```
Loop:   beq  $9, $0, End
        add  $8, $8, $10
        addi $9, $9, -1
        j    Loop
End:
```



```
Text Segment
Bkpt  Address     Code        Basic                          Source
      0x00400000  0x11200003  beq $9,$0,0x00000003           1: Loop: beq $9, $0, End
      0x00400004  0x010a4020  add $8,$8,$10                  2:       add $8, $8, $10
      0x00400008  0x2129ffff  addi $9,$9,0xffffffff          3:       addi $9, $9, -1
      0x0040000c  0x08100000  j 0x00400000                   4:       j Loop

Labels
Label         Address ▲
   mips1.asm
Loop          0x00400000
End           0x00400010

-- program is finished running (dropped off bottom) --
```

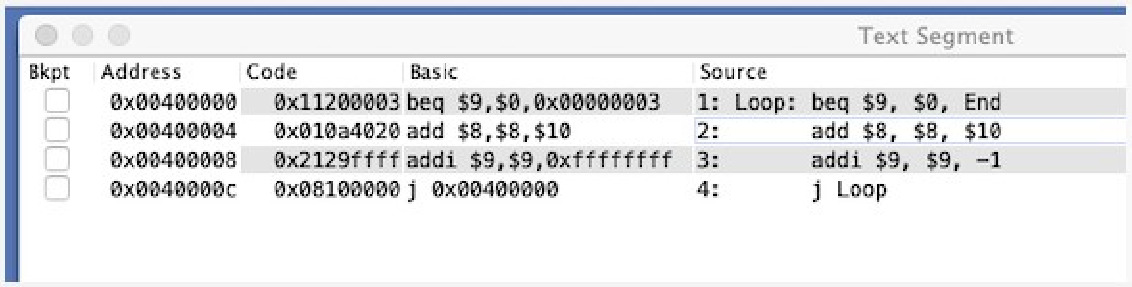
4. Run -> assembler -> go and check out the encoding of th

5. Find out the Program Counter value, the initial address of the first instruction and the j Loop instruction. Explain why the j Loop is coded in the specified Hexadecimal code and the basic format.

e.g. in t

in code

in basic



```
Text Segment
Bkpt  Address     Code        Basic                          Source
      0x00400000  0x11200003  beq $9,$0,0x00000003           1: Loop: beq $9, $0, End
      0x00400004  0x010a4020  add $8,$8,$10                  2:       add $8, $8, $10
      0x00400008  0x2129ffff  addi $9,$9,0xffffffff          3:       addi $9, $9, -1
      0x0040000c  0x08100000  j 0x00400000                   4:       j Loop
```

Note that when you run yours, you might have different code and basic encoding pattern.

Explain your reasonings based on your understanding of MIPS J instruction encoding format.

Submit your answers in detail to BB and including all your team member names and their roles and responsibilities in this Part 3 deliverable.

By completing Part 3, you will be confident to move on to Part 4 - Single Cycle Processor Code Walkthrough.

REASONINGS/EXPLANATION:

For the program counter it's value initially is 0x00400000 and when we either step or run through the program, after execution the value of the program counter becomes 0x00400010 because the first instruction will branch to the end of the program if the values inside of register number $0 and register number $9 are the same value, and this case they are because we did not give the program any values and since the registers are equal to '0' they will branch to the address 0x00000003, the end of the program. The initial address of the first instruction is 0x00400000, where the location of the branch if equal to instruction is. This instruction is the first instruction that is executed by the program when it is ran. The initial address of the j loop instruction would be '0x00400000c' in hexadecimal, and the instruction for the j loop itself is also stored at this address. The j loop instruction is coded in the designated format for the hexadecimal code as a result of the following:

1. Given the basic format for the j instruction: j 0x00400000 which we can convert to the given binary version-

000010	0000 0000 0100 0000 0000 0000 0000 0000

opcode(for j instruction) 32-bit target address

2. When converting the j instruction format in basic format to code format we need to convert the 32-bit basic format target address to a 26-bit format string which is equal to the given hexadecimal value in code format.
Given the string basic format : 0000 1000 0000 0001 0000 0000 0000 0000

0000 0100 0000 0000 0000 0000 00

^
26-bit target address

Then we remove the four bits after the opcode, in this case the following bits are then removed

0000 1000 0000 0001 0000 0000 0000 0000

➔ Then we are left with the following bit string:

0000 1000 0001 0000 0000 0000 0000

Then we remove the 2 bits for the target address word address of which is given by the following:

0000 1000 0001 0000 0000 0000 0000

0000 1000 0001 0000 0000 0000 00

= 26 bit string

→

0 8 1 0 0 0 0 = equivalent hexadecimal representation