CS 241, Lecture 3 - More on Machine Language

1 Addition examples

- Add 11 and 13, return in register 3.

```
1 lis $1
2 .word 11
3 lis $2
4 .word 13
5 add $3, $1, $2
6 jr $31
```

2 Multplication and Division

- We get a problem with multiplying and division - we may need more space when multiplying (ie: $2^{30} \times 2^{30} = 2^{60}$), and when dividing, we want the quotient and remainder!
- For multiplication, we COMBINE the hi and lo registers to get a 64-bit register. The most significant word is placed in hi, and least significant word in lo (most sig word is largest 4 bytes).
- mult $s, $t.
- For division, we put the quotient $s $\div$ $t in lo and the remainder $s $\%$ $t in hi. Preforms s / t
- To access data from hi and lo, we use the mfhi $d and mflo $d instructions to move from the hi/lo register to the register we state.

3 RAM

- RAM uses DRAM which is cheaper but slower than SRAM (so slower than register but more storage than SRAM due to it being cheaper).
- Connect RAM and CPU via bus.

- Starts from address 0 and increments by the word size (4).
- We cannot use data straight from RAM, must transfer to registers first before operations can be done.
- We use the command lw $t, i($s) to load the word in $s + i and stores it in the memory address $t.
- i is an immediate; it is an integer, twos-complement number converted to binary.
- We use sw to do the opposite.
- For example, suppose $1 contains the address of an array and $2 contains the number of elements in the array. Place the immediate value 7 in the last possible spot in the array.

```
1 lis $3
2 .word 4
3 mult $3, $2
4 lis $4
5 .word 7
6 mflo $6
7 sw 4-4(\6)
```

4 Branching

- Use the beq $s, $t, i or bne (same arguments) to branch if the two registers are equal or not equal, respectively.
- We skip our PC by $i \cdot 4$ bytes, or i words, forward.
- For example, place the value 1 in register 2 if register 1 is even, and place 0 in register 2 otherwise.

```
1 lis $3
2 .word 1
3 lis $4
4 .word 2
5 div $1 $5
```

```
 6  mfhi $4
 7  add $2, $0, $0
 8  beq $0, $4, 1
 9  add $2, $0, $3
10  jr $31
```

- Note that branching assumes you already incremented by 4. Remember 251 - we increment by 4 in parallel with the instruction when it goes into the instruction memory stage.
- slt d s t instruction will set d to be 1 if s ¡ t, if s ¿= t then it sets it to be 0.
- If we were to redo the above example using just slt (got lazy, didn't add register values, doesn't matter as no immediate values):

```
1  lis 3
2  .word 1
3  lis 4
4  .word 2
5  div 1 5
6  mfhi 4
7  slt 2 3 4 ; if 4 >= 3, then 4 is 1, so set 2 to 0.
       Othwerise, if 4 < 3, then 4 must be a 0 (even
     ) and so set 2 to 1.
8  jr 31
```

- Example: write a program that negates the value in reg 1 if it is positive.

```
1  slt 2 1 0
2  bne 2 0 1
3  sub 1 0 1
4  jr 31
```

-

4.1 Looping

- Example: add all even numbers from 1 to 20, store in register 3.

```
1  lis 1
2  .word 0x20
3  lis 2
4  .word 0x2
5  add 3 0 0
6  add 3 3 1
7  sub 1 1 2
8  beq 1 0 -3
9  jr 31
```

- We can use labels to make the loops less awful and more readable:

```
1  lis 2
2  .word 20
3  lis 1
4  add 3, 0, 0
5  top:
6      add 3, 3, 2
7      sub 2, 2, 1
8      bne 2, 0, top
9      jr 31
```