Test 2 Review

Brandon Thompson

November 6, 2019

1. Convert between machine code and assembly.
   `0x2237FFF1`
   Convert to binary → `0010|0010|0011|0111|1111|1111|1111|0001`
   separate into field values `001000|10001|10111|1111 1111 1111 0001`
   get machine code from field values: op = 8, rs = 17, rt = 23, imm = -15
   Instruction is: `addi $s7, $s1, -15`

   `0x02F34022`
   Convert to binary → `0000|0010|1111|0011|0100|0000|0010|0010`
   Separate into field values `000000|10111|10011|01000|00000|100010`
   Get machine code from field values: op = 0, rs =23, rt = 19, rd = 8, shamt = 0, funct = 34
   Instruction: `sub $t0,$s7,$s3`

   Instruction `lw $t2, 32($0)`
   Field values: op = 35, rs = 0, rt = 10, imm = 32
   Convert to binary field values: `100011|00000|01010|0000 0000 0010 0000`
   Separate to bytes: `1000|1100|0000|1010|0000|0000|0010|0000`
   Convert to hexadecimal: `0x8C0A0020`

2. Convert between high level and assembly.

   - Loops `beq blt slt j jal jr`
   - Arrays: character, integer, . . .
   - Functions: 4+ arguments, 64 bit returns, stack pointer, Recursion
     Convert from C to MIPS assembly:
     C Code

     ```
     int array[1000];
     int i;

     for (i = 0; i < 1000; i++)
         array[i] = array[i] + 8;
     ```

     MIPS Assembly Code

     ```
     # $s0 = array base address, $s1 = i
     lui $s0, 0x2300
     ori $s0, $s0, 0xF000
     addi $s1, $0, 0
     addi $t2, $0, 1000

     loop:
     slt $t0, $s1, $t2
     .
     .
     .
     ```

     C Code

```
int factorial(int n) {
  if (n <= 1)
    return 1;
  else
    return (n * factorial(n-1));
```

MIPS Code

```
factorial:
addi $sp, $sp, -8    # make room
sw $a0, 4($sp)       # stores $a0
sw $ra, 0($sp)       # stores $ra
addi $t0, $0, 2
slt $t0, $a0, $t0    # n <= 1
.
.
.
```

3. Characters use `lb` (load byte) instruction instead of `lw` (load word) instruction.
   C Code

```
int array[5];
array[0] = array[0] = 2;
array[1] = array[1] = 2;
```

   MIPS Code

```
lui #s0, 0x1234
ori $s0, $a0, 0x8000

lw $t1, 0($a0)
sll $t1, $t1, 1
sw $t1, 0($a0)
.
.
.
```

4. Addressing Modes

   - Base Addressing
     Address of operand is: `base address`
   - PC-Relative Addressing
   - Pseudo-direct Addressing
     Take 32-bit address and make it a 26-bit address by removeing the first 4 and last 2 bits