MIPS has “R-type” and “I-type” instructions that are 32 bits. Assume $rX$ means register $X$. Show the hex machine code equivalent for the following instructions.

The op code for R-type is always zero, and the function code for addition is 32 (decimal).

The op code for add immediate is 8 (decimal).

The op code for load word is 35 (decimal), while the op code for store word is 43 (decimal).

Hex equivalent?

```
add $r17, $r18, $r16
addi $r17, $r16, 5
addi $r19, $r8, -12
lw $r10, 32($r0)
sw $r9, 4($r17)
```

You might want to look at the slides from chapter 6 if you’re having trouble here….

Knowing what you know about machine code, what is the assembly language equivalent of 0x22300005?

Convert the following C code into the MIPS assembly language equivalent

```
if (a0 != a1)
    a2 = a2 + 5;

if (a0 != a1)
{
    if (a0 != a3)
        a2 = a2 + 5;
    else
        a2 = a1 + 6;
}
For the following questions, assume that the data segment starts at 0x10010000, and any code starts at 0x40000000. The MIPS assembly language sometimes converts a single machine instruction into two simpler instructions; go ahead and assume that this does not happen, and each instruction is only 32 bits.

```mips
.data
a:    .word 10
b:    .word 5
c:    .word 8
d:    .word 3
.word 0
```

What is in `a1`, after this instruction?
```
lw $a1, c
```

What is in `a1`, after this instruction?
```
la $a1, c
```

What is in `t1`, after this instruction?
```
la $a1, c
addi $a1, $a1, -8
lw $t1, 4($a1)
```

Write MIPS assembly code to add up the values in an array that is terminated by zero. You can assume that register `a0` has a pointer to the start of the array. Your result should be in register `v0` when you're done.

Write MIPS assembly code to add up the numbers from 1 to 100 (inclusive). Your result should be in register `v0` when you're done. Yes, I know there’s a closed form solution for this. Write the loop, because that’s what I’m trying to test on this exam!
Write a MIPS assembly subroutine that does the equivalent of this. You should know how to send arguments into the subroutine, and where the result should be to send it back.

```c
int elvis(int a, int b)
{
    return a * 2 + b;
}
```

Write a MIPS assembly to call your subroutine, with the C code....

```c
s0 = elvis(31, 29);
```

You sometimes use a machine instruction called “jal”. If the jal you used above was at memory location 0x40001048, which register was changed, and what value was put into that register when the jal instruction was executed?

Write some assembly code to find the maximum value in an array. Assume that a pointer to the array was passed in register a0, and all values in the array are greater than zero (with zero being used as a sentinel value to indicate the end of the array).
Write MIPS code to push register a0, s0, and ra on the stack.

Write MIPS code to pop registers a0, s0, and ra off the stack.

In the following two chunks of code... what is printed for A and B in each?

```c
void setvalues(int *a, int *b)
{
    *a = *a + 10;
    *b = *b + 20;
}

int main()
{
    int a = 5;
    int b = 6;
    setvalues(&b, &a);
    printf("A is %d and B is %d\n", a, b);
}
```

```c
void setvalues(int a, int b)
{
    a = a + 10;
    b = b + 20;
}

int main()
{
    int a = 5;
    int b = 6;
    setvalues(b, a);
    printf("A is %d and B is %d\n", a, b);
}
```

Trivia questions!
How many registers does a MIPS processor have?
When you make a recursive function, which register must you save on the stack?
Which registers are used to pass arguments to a subroutine?
Which registers hold return values from a subroutine?

MIPS guru question!
la breaks down into two simpler instructions. If we have “la $s0, c”, with “c” being the label from the “.data” section on the prior page, what instructions will actually be executed? What value is in s0 once they’re done?