COSC230-SP2023 MIDTERM 1 KEY

I. Numbers and Bases

1. Convert 0o21 into decimal. **(10 points)**

```
2 * 8 + 1 * 1 = 16 + 1 = 17
```

2. Convert 0x3f24 into binary. **(10 points)**

```
0x3f24 = 0011_1111_0010_0100
```

3. Convert 182 into binary. **(10 points)**

```
182 = 128 + 32 + 16 + 4 + 2 = 0b1011_0110
```

II. RISC-V Instructions

Write the assembly instruction for the given C++ code.

1. t0 = sp – 4; **(5 points)**

```
addi t0, sp, -4
```

2. a0 = *(t0 + 4); [Assume t0 is an unsigned short] **(5 points)**

```
lhu a0, 8(t0)
```

3. s0[4] = gp; [Assume s0 is a signed, 64-bit integer] **(5 points)**

```
sd gp, 32(s0)
```

4. myfunc(1, 2); [Assume the parameters to myfunc are both int64_t] **(5 points)**

```
li a0, 1
li a1, 2
call myfunc
```

III. RISC-V ABI

1. The stack pointer must always be a multiple of what value? **(5 points)**

16 bytes

2. Write the instruction(s) to allocate and store the RA register on the stack. **(10 points)**

```
somefunc:
# Write the instructions below

    addi  sp, sp, -16
    sd    ra, 0(sp)
```

IV. N.O.S.

1. Write a NOS table for the following structure. Make sure you indicate the name, offset, size, and total size of the structure. **(10 points)**

```
struct MyStruct {
    uint8_t op;
    int32_t val1;
    int16_t val2;
};
```

Name	Offset	Size
op	0	1
val1	4	4
val2	8	2

Total size: __12__ bytes.

V. Assembly

1. Convert the following C++ into assembly using the MyStruct structure above. **(25 points)**

```
long dofunc(const MyStruct &ms) {
   if (ms.op) return ms.val1 / ms.val2;
   else return ms.val1 % ms.val2;
}
```

```
.section .text
.global dofunc
dofunc:
   # a0 – const MyStruct &ms
   # t0 – ms.op
   # t1 – ms.val1
   # t2 – ms.val2
   lbu    t0, 0(a0)
   lw     t1, 4(a0)
   lh     t2, 8(a0)

   beqz   t0, 2f

   # If we get here, ms.op != 0, we need /
   div    a0, t1, t2
   ret
2:
   # If we get here, ms.op == 0, we need %
   rem    a0, t1, t2
   ret
```