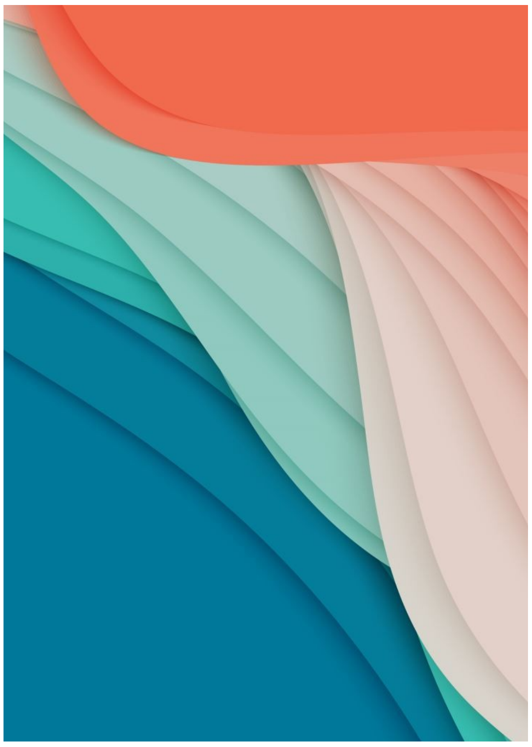

WEEK 12 REVIEW

QUESTION 1

```
typedef struct
{
  char a[5];
  short b[2];
  int c;
} Foo;

void func()
{
  Foo foo;

  foo.a[0] = 1;
  foo.a[1] = 2;
  foo.b[0] = 3;
  foo.c = 4;
}
```

```
func:
# Make space on the stack for
struct
[a]
# foo.a[0] = 1
addi $t0, $zero, 1
[b]
# foo.a[1] = 2
addi $t0, $zero, 2
[c]
# foo.b[0] = 3
addi $t0, $zero, 3
[d]
# foo.c = 4
addi $t0, $zero, 4
[e]
jr $ra
```

```
a. addi $sp, $sp, -16
b. sb $t0, 0($sp)
c. sb $t0, 1($sp)
d. sh $t0, 6($sp)
e. sw $t0, 12($sp)
```

QUESTION 2 – PART A

- We want to save hardware costs by removing the **barrel shifter** from the MIPS ALU.
- Choose an example of an affected instruction:

a) `sllv`

b) `mult`

c) `sub`

d) `addi`

e) `xor`

QUESTION 2 – PART B

Can we still run existing machine code transparently by changing the control unit and adding small amounts of hardware to the datapath (such as a register or a mux)? Note "transparently" means we can execute existing programs on our modified hardware without needing to change the machine code.

Yes, multiply by 2 can replace “`sllv`” but may take more cycles.

QUESTION 3 – PART A

We want to save hardware costs by removing **RAM (main memory)** from the MIPS datapath.

- Choose an example of an affected instruction:

a) `sllv`

b) `mult`

c) `sub`

d) `lbu`

e) `jal`

QUESTION 3 – PART B

Can we still run existing machine code transparently by changing the control unit and adding small amounts of hardware to the datapath (such as a register or a mux)? Note "transparently" means we can execute existing programs on our modified hardware without needing to change the machine code.

No

QUESTION 4 – PART A

We want to save hardware costs by removing the **mutiplier** from the MIPS ALU.

Choose an example of an affected instruction:

a) `sllv`

b) `mult`

c) `sub`

d) `lbu`

e) `jal`

QUESTION 4 – PART B

Can we still run existing machine code transparently by changing the control unit and adding small amounts of hardware to the datapath (such as a register or a mux)? Note "transparently" means we can execute existing programs on our modified hardware without needing to change the machine code.

Yes

QUESTION 5

```
def gcd(x,y):
    if y == 0:
        return x
    else:
        return gcd(y, x % y)
```

QUESTION 5

```
gcd:
    lw $t1, 0($sp)
    addi $sp, $sp, 4
    lw $t0, 0($sp)
    addi $sp, $sp, 4

    bne $t1, $zero, recurse
    addi $sp, $sp, -4
    sw $t0, 0($sp)
    jr $ra

recurse:
    addi $sp, $sp, -4
    sw $AA, 0($sp)

    div $t0, $t1
    mfhi $t2

    addi $sp, $sp, -4
    sw $BB, 0($sp)
    addi $sp, $sp, -4
    sw $t2, 0($sp)

    jal gcd

    lw $t0, 0($sp)
    addi $sp, $sp, 4

    lw $ra, 0($sp)
    addi $sp, $sp, 4

    addi $sp, $sp, -4
    sw $t0, 0($sp)

    jr $ra
```

Below is an assembly implementation of the above. What should the registers in **$AA** **and** **$BB** be?

$AA - $ra

$BB - $t1