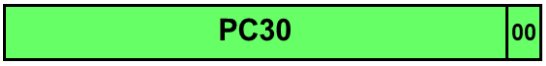
Quiz 4 Solution, COE 233, Term 232
Dr. Muhammad Elrabaa

Q1) This question is about Data & Instruction alignment in Memory: Specify whether each statement below it is **T**rue or **F**alse and give a very brief justification of your answer: (MIPS text segment starts at address **0x04000000** and Data segment at **0x10000000**) **[6 points]**

a. **0x04F0000A** is a valid instruction address **(T/F) Why?:**

False! Instructions are words (i.e. 32-bits) – their address must be word-aligned, i.e. divisible by 4, i.e. ends with 00 .. that is why the PC is only 30 bits, the last two bits are hard-wired to 00



b. **0x14F0000A** is a valid .BYTE data address **(T/F) Why?:**

0x10000000 is the start of the data segment □ Any address ≥ 0x10000000 is a valid BYTE data address (memory addresses are byte aligned anyways, because the memory is byte addressable (i.e. we can put a byte in any address)

c. **0x14F0000A** is a valid .WORD data address **(T/F) Why?:**

Though it is within the data segment (address ≥ 0x10000000), it is not word-aligned (it ends in 10, not 00)

Q2) Complete the code below for an assembly program that reads two integers from the input, put them in registers **$t1, $t2** then compute **$t3 = 5*$t1 + $t2 − 25** and store the value of **$t3** in memory word location **DATA** (already declared in the .DATA segment)**.** Do not use Pseudo instructions. **[6 points]**

```
li  $v0, 5          # Read 1st integer – 5 is the system call number that reads integers from input
syscall
or $t1, $v0, $0     # move the 1st read integer to $t1 .. could also use the pseudo instruction move $t1, $v0
li  $v0, 5          # Read 2nd integer
syscall
or $t2, $v0, $0     # move the 2nd read integer to $t2 .. could also use move $t2, $v0
sll $t6, $t1, 2     # shift $t1 left by 2 bits □ $t6 now will have 4*$t1 ..
addu $t1, $t1, $t6  # add $t6 to $t1 □ $t1 now will have 5*$t1 ..
addu $t3, $t1, $t2  # add $t2 to $t1 □ $t3 now has 5*$t1 + $t2 ..
addi $t3, $t3, -5   # add immediate value -5 to $t3 □ $t3 now has 5*$t1 + $t2 – 5
la $t6, DATA        # load the address of DATA into $t6 – this is a pseudo instruction that loads the
sw $t3, 0($t6)      # address of any label into a register – see slides 37-39 in unit 06, and slide 10 in
                    # unite 07 .. then we store the content of $t3 in memory location stored in $t6
```