University of California, Santa Barbara
Dept. of Computer Science

Practice Questions (and Answers) for Midterm Exam
CS 154, Winter 2020, Matni

IMPORTANT NOTE: These questions are NOT representative of EVERYTHING you need to study for the midterm exam! You should also review your lab assignments questions and all class materials (lecture slides), including the examples and demos. The textbook is a good source of extra material/questions.

1. Convert the following decimal numbers into signed binary. Give answers in two-digit hexadecimals.
   a. 39
   b. 104
   c. -59
   d. -98

2. What is the die yield of a silicon wafer with 800 dies when:
   a. 88 of them do not work?
   b. 700 of them do work?
   c. 15% of them fail initial tests and then 15% of the remaining ones fail secondary tests?

3. A CPU runs at ***X***. If you are told that the average CPI is ***Y***, and that the number of instructions in a program is ***Z***, then how fast will this CPU execute this program, given that:
   a. X = 1.0 GHz, Y = 2.2, Z = 1,000
   b. X = 500 MHz, Y = 1.3, Z = 1,000,000
   c. X = 5.0 GHz, Y = 1.0, Z = 1,000,000

4. If you want to improve a partial design within your CPU by a factor of **5**, then by how much do you improve the entire CPU performance? Given that the improved part of your CPU used to provide 80% of your entire CPU performance.

5. Can I issue the MIPS command **sw $t0, 0x10004320**? Why or why not?

6. What are the MIPS instructions (in hex) for:
   a. `nor $t0, $s1, $s2`
   b. `addi $v0, $t4, -77`
   c. `lw $t1, 4($t2)`
   d. `jal L1` (where L1 is 7 instructions *above* this instruction)

7. Write this C/C++ code in MIPS assembly without using pseudocodes (except for **la**):

```
int a[6] = {-1, -2, 3, 4, -5, 6}, size = 6;
for (int j = 0; j < size; j++)
      a[j] = a[j]*4;
```

8. What would have to happen to the I-type instructions if the MIPS architecture were re-designed to have 16 registers instead of 32, but the opcodes were kept exactly as they were and the MIPS instructions remained 32 bits long?

9. Walk through the MIPS arithmetic algorithm of integer multiplication example of M*N where M = 33 and N = 10.

10. What is the decimal number expressed in these IEEE-754 single-precision floating point numbers?
    a. 0x43800000 256.0
    b. 0xC3002000 -128.125
    c. 0xBEE00000 -0.4375

11. Express the IEEE-754 single-precision floating point number for:
    a. 12.25
    b. -0.625
    c. 11.5
    d. 160.5 5

ANSWERS:

1.
    a. 0x27
    b. 0x68
    c. 0xC5
    d. 0x9E

2.
    a. 800 – 88 = 712, so yield = 712/800 = **0.89 or 89%**
    b. yield = 700/800 = **0.875 or 87.5%**
    c. Initial tests: 120 failures, secondary tests: 102 failures ((800-120)*0.15), so yield = (800 – 222)/800 = 578/800 = **0.7225 or 72.25%**

3. Since CPU time = CPI x IC / Clock Rate = Y.Z/X, so:
    a. time = $2.2x10^3/10^9 = 2.2 \times 10^{-6}$ seconds, or **2.2 μs.**
    b. time = $1.3x10^6/0.5x10^9 = 2.6 \times 10^{-3}$, or **26 ms.**
    c. time = $10^6/5.0x10^9 = 0.2 \times 10^{-3}$, or **200 μs.**

4. $T_{improved} = T_{affected}/5 + T_{unaffected} = 80/5 + 20 = 16 + 20 = 36$.
   So, improvement factor is 100/36 = ~ **2.78**.

5. The MIPS memory address **0x10004320** is in the designated Static Data region and adheres to the big-endian style of MIPS memory addressing (the address is a multiple of 4). Therefore, saving a word at that memory address would work.

6.
   a. 0x02324027
   b. 0x2182FFB3
   c. 0x8D490004
   d. 0x0FFFFFF9, since -7 is 0x3FFFFF9 in 26 bits signed and jal is 0x03 in 6 bits.

7.
```
.data
      a: .word -1 -2 3 4 -5 6

.text
main:
      addi $t0, $zero, 6    # var size
      addi $t1, $zero, 0    # var j
      la $t2, a             # var &a
loop: lw $t3, 0($t2)
      sll $t3, $t3, 2
      sw $t3, 0($t2)
      beq $t1, $t0, exit
      addi $t2, $t2, 4
      addi $t1, $t1, 1
      j loop
exit: addi $v0, $zero, 10
      syscall
```

8. If we now have 16 registers instead of 32, then it would be best to change the **rs** and **rt** fields to go from 5 bits to 4 bits (because $2^4 = 16$), although that isn't strictly necessary since 5 bits can still accommodate 16 registers. However, if we do shrink the **rs**/**rt** fields and kept the **opcode** as it were and kept the instructions at 32 bits, then the **immediate** field would grow from 16 bits to 18 bits. This would mean that we could expand the range of *integer* numbers we can utilize by a factor of 4 since the range would grow from $[-2^{15} , 2^{15}-1]$ to $[-2^{17} , 2^{17}-1]$.

9.
   1. M = 33 = 0x00000021 = 0…100001, N = 10 = 0x0000000A = 0…1010
   2. Partial product P = 0
   3. N[0] = 0 ➔ P = 0
   4. sra N by 1 so N = 0…101
   5. sll M by 1 so M = 0…1000010
   6. N[0] = 1 ➔ P += M = 0…1000010
   7. sra N by 1 so N = 0…10
   8. sll M by 1 so M = 0…10000100
   9. N[0] = 0 ➔ P = 0…1000010 (unchanged)
   10. sra N by 1 so N = 0…1
   11. sll M by 1 so M = 0…100001000
   12. N[0] = 1 ➔ P += M = 0…1000010 + 0…100001000 = 101001010
   13. sra N by 1 so N = 0 ➔ exit loop
   14. Final result is P = M*N = 101001010 = 330 in decimal

10.
    a. 256.0
    b. -128.125
    c. -0.4375
11.
    a. 0x41440000
    b. 0xbf200000
    c. 0x41380000
    d. 0x43208000