Exam 1 Review Problem Solutions

1. Add a line that says ".align 2" right before the ".space 4" directive. The problem is that x is not aligned on a byte address that is a multiple of 4.

2. 1111111111111111 is -1 in base 10.

3. 7af3 is 0111 1010 1111 0011 in binary.

4. 43 is $32 + 8 + 2 + 1 = 0000 0000 0010 1011$ in binary = 002B in hexadecimal.

5. Convert -43 to binary by flipping bits and adding 1:

```
1111 1111 1101 0100
+1
---------
1111 1111 1101 0101
```

which is FFD5 in hexadecimal.

6. #include <stdio.h>
   int main() {
      int c;
      while ((c = getchar()) != '\n') {
         if (c >= 'a' && c <= 'z') {
            c = c - 'a' + 'A';
         }
         putchar(c);
      }
      putchar('\n');
   }

7. If $I$ = number of instructions, $CPI$ = number of cycles per instruction, and $T$ = time needed for one cycle, then the performance equation describes the time needed to execute $I$ instructions:

$$I_{ins} \times CPI_{cyc} \frac{cyc}{ins} \times T_{sec} \frac{sec}{cyc}$$

8. The ratio of A’s running time to B’s running time is:

$$\frac{3 \times 10^{-6}}{0.5 \times 10^{-6}} = 6$$

so processor B is 6 faster/slower than A.
9. For a fixed number of instructions $I$ we want the execution time of $A$ to equal the execution time of $B$:

\[ I_{\text{ins}} \times \frac{4}{15} \times \frac{1}{2 \times 10^9} \text{cyc} = I_{\text{ins}} \times x \times \frac{1}{1.5 \times 10^9} \text{cyc} \]

so $x = 2 \times 1.5 = 3 \times \frac{\text{cyc}}{\text{ins}}$.

10. lw $t0,x$
    lw $t1,y$
    add $t0,$t0,$t1
    sw $t0,z$

11. lw $t0,x$
    lw $t1,y$
    beq $t0,$t1,eq
    sw $zero,z$
    j done
    eq: li $t0,30$
    sw $t0,z$
    done: ...

12. The output will be “this i” (the first six characters in string s).

13. Here is the instruction translated into binary, followed by the same number split into the groupings required by the instruction format:

```
0000 0000 0100 0011 0010 0000 0010 0000 0000 0010 0000 0000 00010 00011 00100 00000 100000
```

Reading the second, third, and fourth groups, we see that the registers are rs = 2, rt = 3, and rd = 4. (These numbers correspond to registers $v0$, $v1$, and $a0$, but you weren’t required to know that.)

14. 
   
15. `printf("%d,%d,%d\n", 100, 1, 65, 3.333, 5);`

16. 
\[
\left( 500_{\text{ins}} \times 3 \frac{\text{cyc}}{\text{ins}} + 500_{\text{ins}} \times 2 \frac{\text{cyc}}{\text{ins}} \right) \times 10^{-9} \frac{\text{cyc}}{\text{sec}} = 2500 \times 10^{-9} \text{sec} = 2.5 \times 10^{-6} \text{sec} = 2.5 \mu\text{sec}
\]
17. 

```c
for ((c = getchar()) != EOF) {
    putchar(c);
}
```

18. Since the `slt` instruction fails to set register $t0$ to 1 (because 40 is not less than 30), the following `bne` instruction fails to branch to `label` (the register does, in fact, equal zero). Therefore, the value 15 will be stored in `result`.

19. 

```
$t0 = 1111 1111 1111 1111 1111 1111 1111 1111$
$t1 = 0000 0000 0000 0000 0000 0000 0000 1111$
$t2 = 0000 0000 0000 0000 0000 0000 0000 1111 = 0x000000F
```

20. 

```
$t0 = 1111 1111 1111 1111 1111 1111 1111 1111$
$t1 = 0000 0000 0000 0000 0000 0000 0000 1111$
$t2 = 1111 1111 1111 1111 1111 1111 1111 1111 = 0xFFFFFFFF
```

21. 

```
$t0 = 1111 1111 1111 1111 1111 1111 1111 1111$
$t2 = 1111 1111 1111 1111 1111 1111 1111 1111 = 0xFFFFFFFF
```

Note that in a "shift right arithmetic" the sign bit is copied as the bits on the left are filled in.

The above is not a full review!