Solution

Problem 1: (14 points)

[1]   0000 1101   [2]   1111 1010   [3]   1111 1011
[4]   0000 1101   [5]   0011 0110   [6]   0101 0010
[7]   0000 0000

Problem 2: (12 points)

[1]   %edx   [2]   0x00000001
[3]   %edx   [4]   0x00000042
[5]   0x0000400c   [6]   0x0000000a
[7]   --   [8]   --
[9]   %eax   [10]   0x00004008
[11]   %esp, 0x0000400c   [12]   0x0000400c, 0x00000001

Problem 3: (18 points)

1. [1]   20   [2]   32
   [3]   32   [4]   48
   [5]   32   [6]   48
   [7]   0x804a044   [8]   0x601068
   [9]   0x804a04c   [10]   0x601078
   [11]   0x804a044   [12]   0x601068
   [13]   0x804a054   [14]   0x601080

2. X86: 32 − (1+4+3+2+8+1+4) = 9 bytes wasted
   X86-64: 48 − (1+8+3+2+16+1+4) = 13 bytes wasted

3. X86: 8 bytes. It will use 24 bytes at least.
   (For example, 1 byte padding at the end of struct)

   X86-64: 8 bytes. It will use 40 bytes at least.
   (For example, 5 byte padding at the end of struct)

Problem 4: (9 points)

1 [1]   -30
   [2]   1 111111 00000
   [3]   1 000001 00000

2. $-6.5 = (-1) * (1 + \frac{1}{2} + \frac{1}{8}) * 2^2$

   1 100001 10100

3. 0 000011 10110

Problem 5: (25 points)

1 [1] 'c'   [2] str[i][j]
   [3] 'a'   [4] result << 2
   [5]   NONE   [6] result > j ? 'A' : result
   [7] L7   [8] %ebx
   [9] (%edx,%eax,4)   [10] -4(%ebp)
   [11] $4   [12] *.L6(, %eax, 4)
   [13] -8(%ebp)   [14] -8(%ebp)

2 **%ebx is a callee-saved register, so it should be saved in
   the stack before using it and restored before returning.**

3 **cc is : D**

Problem 6: (22 points)

1 [1] (%eax)   [2] (%eax,%edx,1)
   [3] 0x4(%esp)   [4] 40 85 04 08
   [5] 0xa(%esp)   [6] leave

2 [1] 0xffffcb18   [2] 0xffffcba8
   [3] 0xffffcafc   [4] 0xffffcb18
   [5] 0xffffcad0   [6] 0xffffcaf8
   [7] 0xffffcb00   [8] 0xffffcb18

3 [1] 0x4020100   [2] old *(int *)n + 0x10100
   [3] 0   [4] n[0]
   [5] 0   [6] c

4 **foo:0
   foo:3
   foo:7**