Problem 1: Floating Point (14points)

1. [1] 7 [2] 0 1111 00000000

   [3] 0 1110 11111111 [4] 0 0000 00000001 / 0 0000 00000000

2. $(1\ 1010\ 00111110)_2$

3. $-0.578125 * 2^{-6}$

4. (1) $(0\ 0100\ 00111100)_2 = 1.00111100 * 2^{-3}$

   $(0\ 1001\ 10101001)_2 = 1.10101001 * 2^{2}$

   (2) $(0\ 0100\ 00111100)_2 = 1.00111100 * 2^{-3}$

   $= 0.0000100111100 * 2^{2}$

   (3) $1.10101001 * 2^{2} + 0.0000100111100 * 2^{2}$

   $= 1.101100101110 * 2^{2}$

   $= 1.10110011 * 2^{2}$ (Round-to Even) (or (0 1001 10110011))

Problem 2: X86-64 (14points)

1 [1] -8(%rbp) [2] %ecx [3] %eax [4] -4(%rbp)

[5] %ecx [6] -36(%rbp) [7] -32(%rbp,%rax,4)

Problem 3: Memory Allocation (14points)

1 1) 2nd operation

↓P3						↓P1				↓P4				↓P2							
24/1					24/1	16/1			16/1	16/1			16/1	8/1	8/1	24/0					24/0

2) 4th operation:

↓P3						↓P1				↓P4				↓P5							
24/1					24/1	16/1			16/1	16/1			16/1	24/1					24/1	8/0	8/0

2 P3: 24-9=15, P4: 16-3=13, P5: 24-15=9. 15+13+9 = 37 bytes. (ambiguous)

3 8 bytes (problematic)

Problem 4: Cache (16points)

1. [1] 32 [2] 8 [3] 2 [4] 2

2. [1] 1 [2] Yes [3] Oxcd

   [4] 1 [5] No [6] --

   [7] 1 [8] Yes [9] --

   [10] 1 [11] No [12] --

Problem 5: Linking (26points)

1. [1]03 [2]$0x3

   [3]04 [4]$0x4

2. [1]R_386_32 [2]0x2f [3]R_386_32 [4]R_386_PC32

   [5]0x3a [6]0x15 [7]R_386_PC32 [8]R_386_PC32

   [9].rodata [10] R_386_32 [11] 0x2b [12] .rodata

3. [1] 05 40 a0 04 08 [2] e8 a5 ff ff ff [3] e8 eb fe ff ff

4. [1] 0x08048336 [2] 5

Problem 6: Optimization (16points)

1.
```
   // eliminate unneeded memory references
   int local_min = 100, local_max = -1;
   // local variables for expansion of function
   int elem;
   // reduce procedure calls
   int row_cnt = row_count(p);
   // Failed to combine the two loops won't affect your grade!
   for (int i = 0; i < row_cnt; i++) {
       for (int j = 0; j < 2; j++) {
       // expansion of function in loop
           elem = p->base[i * 2 + j];
           if (elem > local_max) local_max = elem;
           if (elem < local_min) local_min = elem;
       }
   }
   // eliminate unneeded memory references
   *max = local_max; *min = local_min;
```
2.
```
    // Failed to combine the two loops won't affect your grade!
    int local_min = 100, local_max = -1;
    for (int i = 0; i < row_cnt; i++) {
        int elem0, elem1;
        elem0 = p->base[i * 2];
        if (elem0 > local_max) local_max = elem0;
        if (elem0 < local_min) local_min = elem0;
        // loop unrolling
        elem1 = p->base[i * 2 + 1];
        if (elem1 > local_max) local_max = elem1;
        if (elem1 < local_min)   local_min = elem1;
    }
    *max = local_max; *min = local_min;
```

OR

```
int local_min = 100, local_max = -1;
// Treat 2d array as 1d array.
int elem_cnt = row_count(p) * 2;
for (int i = 0; i < elem_cnt; i++) {
   int elem = p->base[i];
   if (elem > local_max) local_max = elem;
   if (elem < local_min) local_min = elem;
}
*max = local_max; *min = local_min;
```