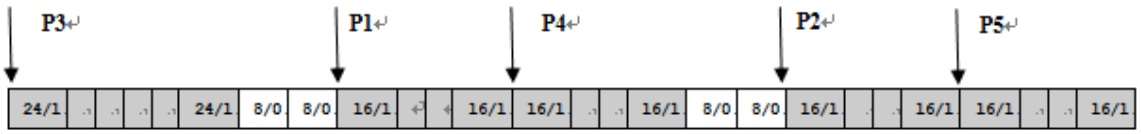
Problem 1: Y86 (13points)

1. [1] 0x006 [2] 2063FF0F [3] 78563412
   [4] irmovl array,%edx [5] 7362000000 [6] popl %ebp

2. [1] 0x12345678 [2] 0xFFFFFFFF [3] 0x20 [4] 0x0
   [5] 0x12345678 [6] 0x100 [7] 0xFC

Problem 2: Memory (12points)

1 1)



2)

3) first-fit. Because first-fit enjoys better memory utilization than best-fit after 7th operation is executed.

Problem 3: x86-64 (20points)

1 [1] %rcx [2] %rdx [3] -0x38(%rbp) [4] (%rsp) [5] %r8d
[6] %rdi [7] %rsi [8] %rcx [9] %r9d [10] 0x18(%rbp)

Problem 4: Cache (27points)

1. [1] 2048 [2] 23 [3] 4 [4] 5

2. [1] 11 [2] Y [3] 11 [4] N
   [5] 10 [6] Y [7] 11 [8] Y
   [9] 10 [10] Y [11] 10 [12] N
   [13] 66.67% (2/3)

3. 1) (1+8+8) * 3 = 51

   2) [(1 + 1) + 1 + 1] / 51 = 7.84% (4/51)

   3) Scheme B

   Reason: The four rows of the array are in set 8, 9, 10, 11 respectively, so increasing the number of sets is useless.
   After doubling the block size, the miss rate is (1 + 1) / 51 = 3.92%.
   Each row of the array only occupies one cache line in a set, so increasing the number of ways is useless.

Problem 5: Linking (28points)

1. [1] 0x13 [2] 0x1d [3] 0x26
   [4] 0x17 [5] 0x32 ([1][2][3]顺序不要求与答案一致，[4][5]也一样)

2. [1] 0xd [2] R_386_32 [3] 0x7
   [4] R_386_32 [5] 0x21 [6] R_386_PC32

3. [1] 8b 15 78 96 04 08
   [2] E8 99 FF FF FF
   [3] C7 05 7C 96 04 08 70 96 04 08

4. 1) 0x080482BE

   2) When printf is first called, corresponding entry in GOT is just a stub to set pc back to the second instruction in PLT entry (0x080482be). After some identifiers are pushed into stack, the dynamic linker will be invoked to patch the actual address of "printf" into the GOT entry of "printf" and call "printf" again.

   If printf is called again, pc will jump to actual address of "printf" after looking up the corresponding entry in GOT.