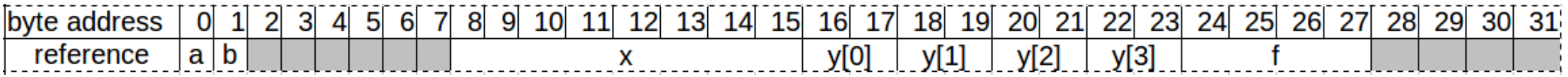
Problem 1

```
struct node_t {
        char a;
        char b;
        long x;
        short y[4];
        float z;
} node_struct;
```

Size = **32 bytes (28 is accepted)**



Problem 2

```
union node_t {
        char a;
        char b;
        long x;
        short y[4];
        float z;
} node_instance;
```

Size = **8 bytes**
(long x = 8 bytes, short y[4] = (2 bytes) * 4 = 8 bytes)

node_instance.z = **-2.75**

Hex(node_instance.z) = 0xc0300000

```
node_instance.y[0] = 0;
node_instance.y[1] = 0xc030;
node_instance.y[2] = 0xffff;
node_instance.y[3] = 0;
```

byte offset address	0	1	2	3	4	5	6	7
contents	0x00	0x00	0x30	0xc0	0xff	0xff	0x00	0x00
union reference	y[0]		y[1]		y[2]		y[3]	
	a							
	b							
	z							
	x							

Problem 3

a) %eax=
Omega: **0x2**
Ying-yang: **0x3**
Infinity: **0x4**
Spades: **0x5**

b) %eax= , %edx=
Omega: **0x3**, **0xc**
Ying-yang: **0x3, 0x8**
Infinity: **0x3, 0x4**
Spades: **0x2, 0xc**

c) %ecx=
Omega: **0x080486f7**
Ying-yang: **0x080486de**
Infinity: **0x080486c4**
Spades: **0x080486f0**

String Output
Spade - **green**
Infinity - **blue**
Yin yang - **grey**
Omega - **onyx**

Problem 3: Omega Solution

```
08048414 <func0>:
 8048414:       55                          push   %ebp
 8048415:       89 e5                       mov    %esp,%ebp
 8048417:       83 ec 18                    sub    $0x18,%esp
 804841a:       8b 45 10                    mov    0x10(%ebp),%eax
 804841d:       83 e8 64                    sub    $0x64,%eax
 8048420:       83 f8 05                    cmp    $0x5,%eax
 8048423:       77 31                       ja     8048456 <func0+0x42>
 8048425:       8b 04 85 80 86 04 08        mov    0x8048680(,%eax,4),%eax
 804842c:       ff e0                       jmp    *%eax
 804842e:       83 45 08 01                 addl   $0x1,0x8(%ebp)
 8048432:       83 45 0c 01                 addl   $0x1,0xc(%ebp)
 8048436:       eb 2c                       jmp    8048464 <func0+0x50>
 8048438:       83 45 08 02                 addl   $0x2,0x8(%ebp)
 804843c:       83 45 0c 02                 addl   $0x2,0xc(%ebp)
 8048440:       eb 22                       jmp    8048464 <func0+0x50>
 8048442:       83 45 08 01                 addl   $0x1,0x8(%ebp)
 8048446:       83 45 0c 02                 addl   $0x2,0xc(%ebp)
 804844a:       eb 18                       jmp    8048464 <func0+0x50>
 804844c:       83 45 08 02                 addl   $0x2,0x8(%ebp)
 8048450:       83 45 0c 01                 addl   $0x1,0xc(%ebp)
 8048454:       eb 0e                       jmp    8048464 <func0+0x50>
 8048456:       c7 45 08 00 00 00 00        movl   $0x0,0x8(%ebp)
 804845d:       c7 45 0c 00 00 00 00        movl   $0x0,0xc(%ebp)
 8048464:       8b 55 08                    mov    0x8(%ebp),%edx
 8048467:       8b 45 0c                    mov    0xc(%ebp),%eax
 804846a:       c1 e2 02                    shl    $0x2,%edx
 804846d:       8d 04 02                    lea    (%edx,%eax,1),%eax
 8048470:       8b 0c 85 e0 98 04 08        mov    0x80498e0(,%eax,4),%ecx
 8048477:       ba 74 86 04 08              mov    $0x8048674,%edx
 804847c:       a1 c0 98 04 08              mov    0x80498c0,%eax
 8048481:       89 4c 24 08                 mov    %ecx,0x8(%esp)
 8048485:       89 54 24 04                 mov    %edx,0x4(%esp)
 8048489:       89 04 24                    mov    %eax,(%esp)
 804848c:       e8 b7 fe ff ff              call   8048348 <fprintf@plt>
 8048491:       8b 45 10                    mov    0x10(%ebp),%eax
 8048494:       c9                          leave
 8048495:       c3                          ret
```

```
(gdb) break *func0
Breakpoint 1 at 0x8048414
```

```
esp            0xffffd3dc
ebp            0xffffd418
```

Problem 3: Omega Solution

```
(gdb) x/32x 0xffffd3dc
0xffffd3dc:   0x08048585   0x00000001   0x00000001   0x00000066
0xffffd3ec:   0x080485d9   0x00b2e1ec   0x0804825d   0x00b30ce0
0xffffd3fc:   0x00b2fff4   0x080485c0   0x08048360   0x00b2fff4
0xffffd40c:   0x00000000   0x080485c0   0x00000000   0xffffd498
0xffffd41c:   0x009b3d26   0x00000004   0xffffd4c4   0xffffd4d8
0xffffd42c:   0xf7ffd428   0x08048360   0xffffffff   0x00999fc4
0xffffd43c:   0x0804825d   0x00000001   0xffffd480   0x00989a45
0xffffd44c:   0x0099aab0   0xf7ffd708   0x00b2fff4   0x00000000

(gdb) x/32x 0x8048680
0x8048680:   0x0804842e   0x08048456   0x08048438   0x08048442
0x8048690:   0x08048446   0x0804844c   0x776f7262   0x70000a6e
0x80486a0:   0x0a6b6e69   0x64657200   0x6877000a   0x0a657469
0x80486b0:   0x616c6200   0x000a6b63   0x61757161   0x6f67000a
0x80486c0:   0x000a646c   0x65756c62   0x636f000a   0x0a657268
0x80486d0:   0x75616d00   0x000a6576   0x6e617963   0x7267000a
0x80486e0:   0x000a7965   0x0a6e6174   0x69656200   0x000a6567
0x80486f0:   0x65657267   0x6f000a6e   0x0a78796e   0x73657400

(gdb) x/16x 0x80498e0
0x80498e0:   0x08048698   0x0804869f   0x080486a5   0x080486aa
0x80498f0:   0x080486b1   0x080486b8   0x080486be   0x080486c4
0x8049900:   0x080486ca   0x080486d1   0x080486d8   0x080486de
0x8049910:   0x080486e4   0x080486e9   0x080486f0   0x080486f7

(gdb) x/128x 0x8048690
0x8048690:   0x08048446   0x0804844c   0x776f7262   0x70000a6e
0x80486a0:   0x0a6b6e69   0x64657200   0x6877000a   0x0a657469
0x80486b0:   0x616c6200   0x000a6b63   0x61757161   0x6f67000a
0x80486c0:   0x000a646c   0x65756c62   0x636f000a   0x0a657268
0x80486d0:   0x75616d00   0x000a6576   0x6e617963   0x7267000a
0x80486e0:   0x000a7965   0x0a6e6174   0x69656200   0x000a6567
0x80486f0:   0x65657267   0x6f000a6e   0x0a78796e   0x73657400
0x8048700:   0x676e6974   0x64252020   0x0000000a   0x3b031b01
```

Problem 3: Omega Solution

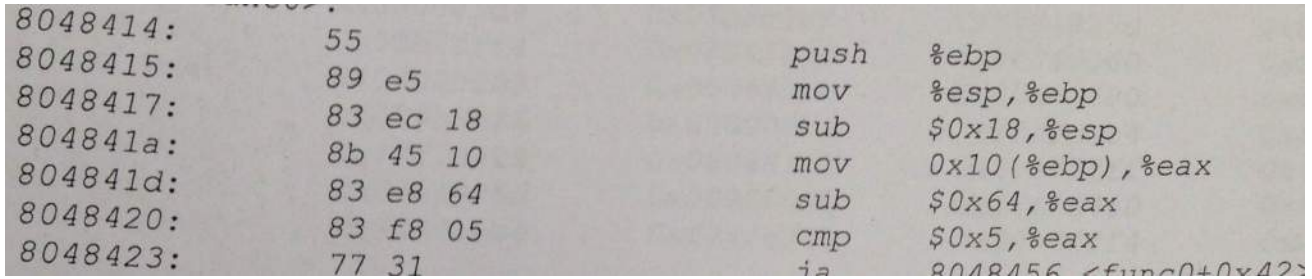
Question 1: Value of %eax and %edx at 0x804846d before execution.

```
                                    sub    $0x64,%eax
8048420:  83 f8 05                  cmp    $0x5,%eax
8048423:  77 31                     ja     8048456 <func0+0x42>
8048425:  8b 04 85 80 86 04 08      mov    0x8048680(,%eax,4),%eax
804842c:  ff e0                     jmp    *%eax
804842e:  83 45 08 01               addl   $0x1,0x8(%ebp)
8048432:  83 45 0c 01               addl   $0x1,0xc(%ebp)
8048436:  eb 2c                     jmp    8048464 <func0+0x50>
8048438:  83 45 08 02               addl   $0x2,0x8(%ebp)
804843c:  83 45 0c 02               addl   $0x2,0xc(%ebp)
8048440:  eb 22                     jmp    8048464 <func0+0x50>
8048442:  83 45 08 01               addl   $0x1,0x8(%ebp)
8048446:  83 45 0c 02               addl   $0x2,0xc(%ebp)
804844a:  eb 18                     jmp    8048464 <func0+0x50>
804844c:  83 45 08 02               addl   $0x2,0x8(%ebp)
8048450:  83 45 0c 01               addl   $0x1,0xc(%ebp)
8048454:  eb 0e                     jmp    8048464 <func0+0x50>
8048456:  c7 45 08 00 00 00 00      movl   $0x0,0x8(%ebp)
804845d:  c7 45 0c 00 00 00 00      movl   $0x0,0xc(%ebp)
8048464:  8b 55 08                  mov    0x8(%ebp),%edx
8048467:  8b 45 0c                  mov    0xc(%ebp),%eax
804846a:  c1 e2 02                  shl    $0x2,%edx
804846d:  8d 04 02                  lea    (%edx,%eax,1),%eax
```

We dont jump
eax = eax*4 + 0x8048680
= 0x8048688

Problem 3: Omega Solution

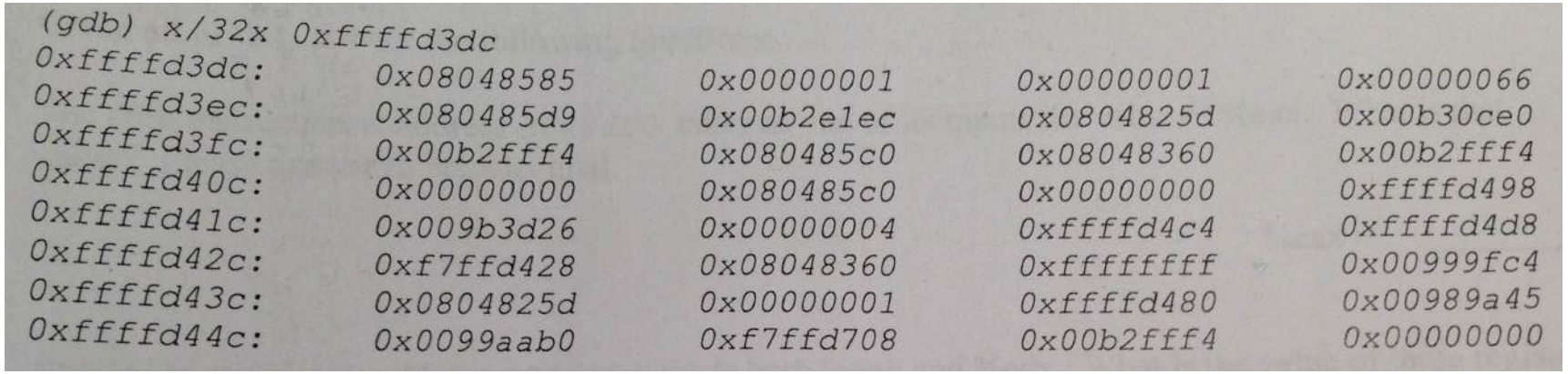
Question 1: Value of %eax at cmp instruction (0x8048420)



```
8048414:       55                          push   %ebp
8048415:       89 e5                       mov    %esp,%ebp
8048417:       83 ec 18                    sub    $0x18,%esp
804841a:       8b 45 10                    mov    0x10(%ebp),%eax
804841d:       83 e8 64                    sub    $0x64,%eax
8048420:       83 f8 05                    cmp    $0x5,%eax
8048423:       77 31                       ja     8048456 <func0+0x42>
```

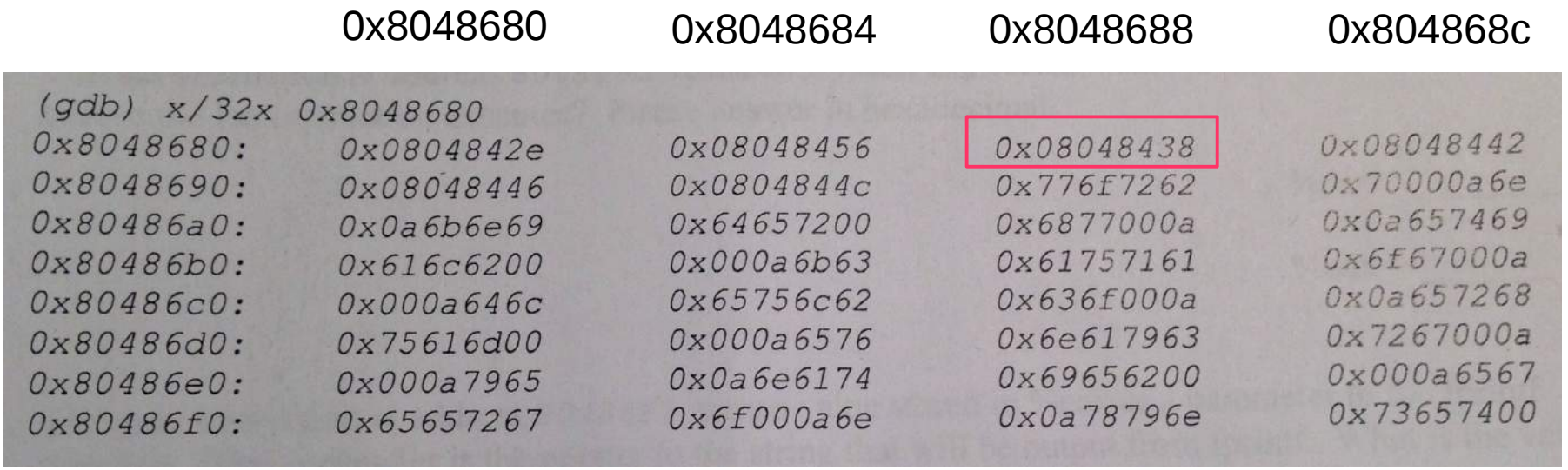
esp = esp – 4 //
esp = 0xffffd3dc – 4
esp = 0xffffd3d8
ebp = esp
0x10(%ebp) = address 0xfffd3d8+0x10
= 0xfffd3e8

0xffffd3dc 0xffffd3e0 0xffffd3e4 **0xffffd3e8**



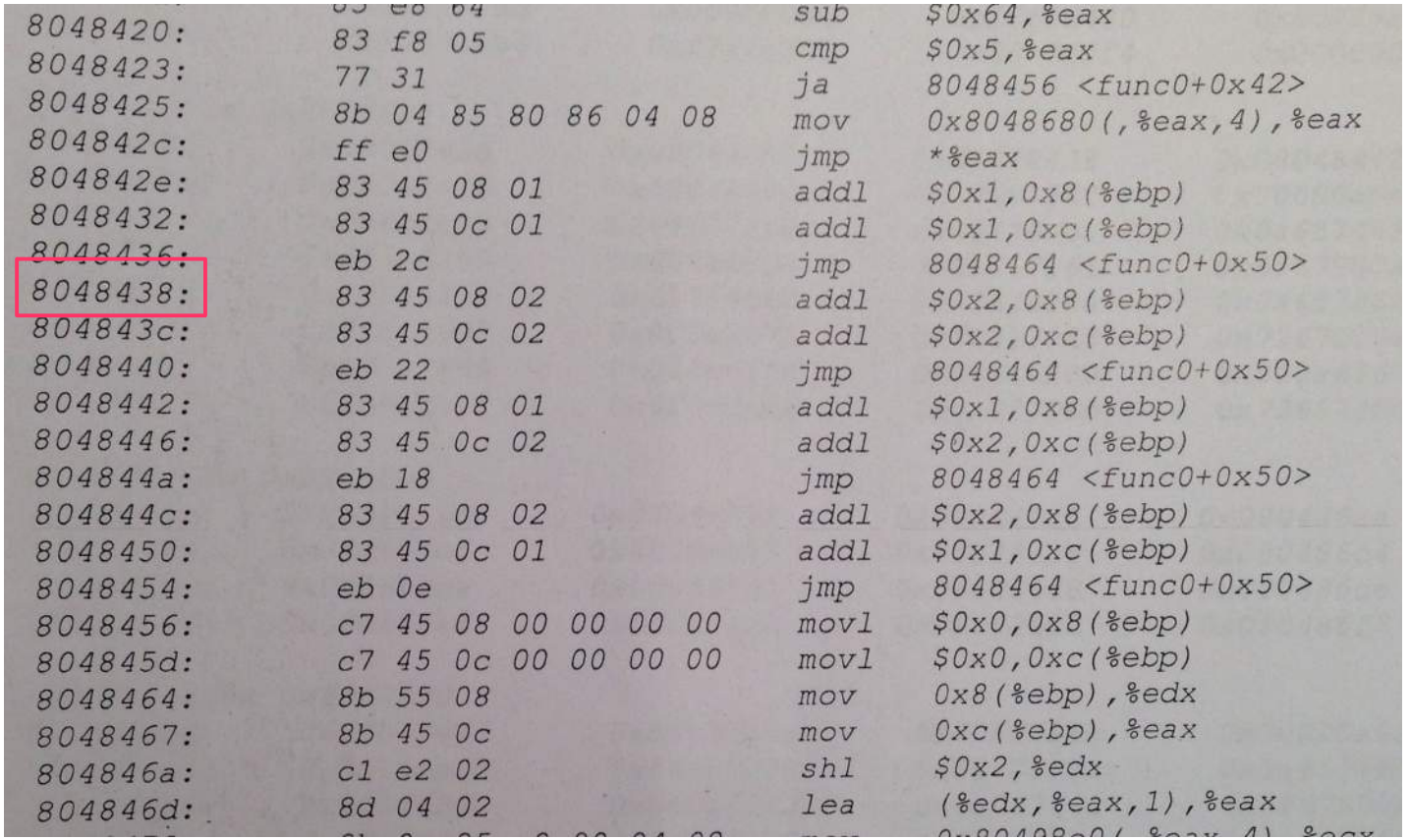
```
(gdb) x/32x 0xffffd3dc
0xffffd3dc:    0x08048585    0x00000001    0x00000001    0x00000066
0xffffd3ec:    0x080485d9    0x00b2e1ec    0x0804825d    0x00b30ce0
0xffffd3fc:    0x00b2fff4    0x080485c0    0x08048360    0x00b2fff4
0xffffd40c:    0x00000000    0x080485c0    0x00000000    0xffffd498
0xffffd41c:    0x009b3d26    0x00000004    0xffffd4c4    0xffffd4d8
0xffffd42c:    0xf7ffd428    0x08048360    0xffffffff    0x00999fc4
0xffffd43c:    0x0804825d    0x00000001    0xffffd480    0x00989a45
0xffffd44c:    0x0099aab0    0xf7ffd708    0x00b2fff4    0x00000000
```

eax = 0x66
0x66 – 0x64
eax = **0x2**


0x8048680
0x8048684
0x8048688
0x804868c
(gdb) x/32x 0x8048680
0x8048680: 0x0804842e 0x08048456 0x08048438 0x08048442
0x8048690: 0x08048446 0x0804844c 0x776f7262 0x70000a6e
0x80486a0: 0x0a6b6e69 0x64657200 0x6877000a 0x0a657469
0x80486b0: 0x616c6200 0x000a6b63 0x61757161 0x6f67000a
0x80486c0: 0x000a646c 0x65756c62 0x636f000a 0x0a657268
0x80486d0: 0x75616d00 0x000a6576 0x6e617963 0x7267000a
0x80486e0: 0x000a7965 0x0a6e6174 0x69656200 0x000a6567
0x80486f0: 0x65657267 0x6f000a6e 0x0a78796e 0x73657400

Problem 3: Omega Solution

Question 1: Value of %eax and %edx at 0x804846d before execution.



```
                                    sub    $0x64,%eax
8048420:  83 f8 05                  cmp    $0x5,%eax
8048423:  77 31                     ja     8048456 <func0+0x42>
8048425:  8b 04 85 80 86 04 08      mov    0x8048680(,%eax,4),%eax
804842c:  ff e0                     jmp    *%eax
804842e:  83 45 08 01               addl   $0x1,0x8(%ebp)
8048432:  83 45 0c 01               addl   $0x1,0xc(%ebp)
8048436:  eb 2c                     jmp    8048464 <func0+0x50>
8048438:  83 45 08 02               addl   $0x2,0x8(%ebp)
804843c:  83 45 0c 02               addl   $0x2,0xc(%ebp)
8048440:  eb 22                     jmp    8048464 <func0+0x50>
8048442:  83 45 08 01               addl   $0x1,0x8(%ebp)
8048446:  83 45 0c 02               addl   $0x2,0xc(%ebp)
804844a:  eb 18                     jmp    8048464 <func0+0x50>
804844c:  83 45 08 02               addl   $0x2,0x8(%ebp)
8048450:  83 45 0c 01               addl   $0x1,0xc(%ebp)
8048454:  eb 0e                     jmp    8048464 <func0+0x50>
8048456:  c7 45 08 00 00 00 00      movl   $0x0,0x8(%ebp)
804845d:  c7 45 0c 00 00 00 00      movl   $0x0,0xc(%ebp)
8048464:  8b 55 08                  mov    0x8(%ebp),%edx
8048467:  8b 45 0c                  mov    0xc(%ebp),%eax
804846a:  c1 e2 02                  shl    $0x2,%edx
804846d:  8d 04 02                  lea    (%edx,%eax,1),%eax
```

We dont jump
eax = eax*4 + 0x8048680
= 0x8048688

Jump to address 0x08048438
Add 2 to locations 0x8(ebp)
Add 2 to locations 0xc(ebp)

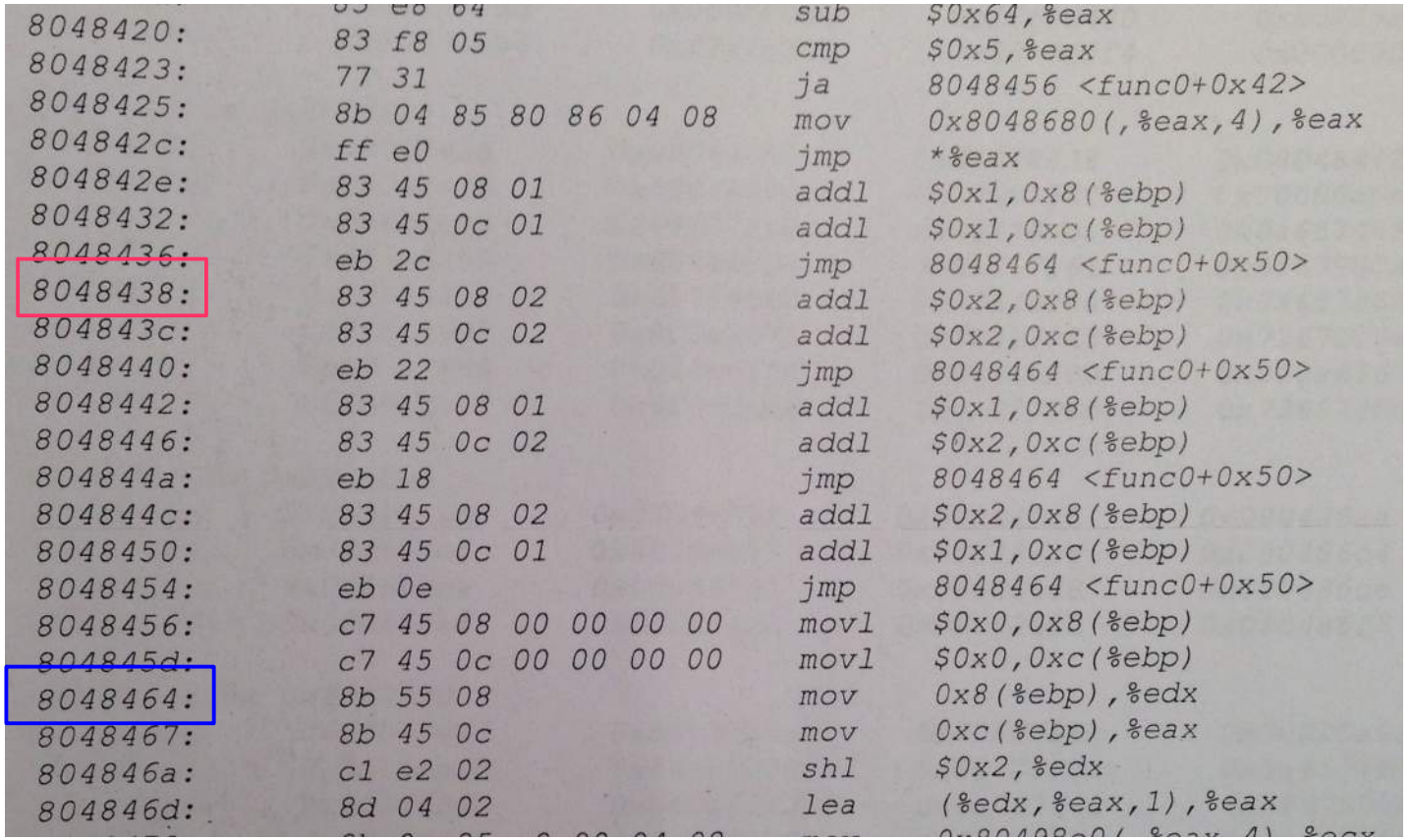
Problem 3: Omega Solution

1 + 2 = **3** 1 + 2 = **3**

```
(gdb) x/32x 0xffffd3dc
0xffffd3dc:   0x08048585   0x00000001   0x00000001   0x00000066
0xffffd3ec:   0x080485d9   0x00b2e1ec   0x0804825d   0x00b30ce0
0xffffd3fc:   0x00b2fff4   0x080485c0   0x08048360   0x00b2fff4
0xffffd40c:   0x00000000   0x080485c0   0x00000000   0xffffd498
0xffffd41c:   0x009b3d26   0x00000004   0xffffd4c4   0xffffd4d8
0xffffd42c:   0xf7ffd428   0x08048360   0xffffffff   0x00999fc4
0xffffd43c:   0x0804825d   0x00000001   0xffffd480   0x00989a45
0xffffd44c:   0x0099aab0   0xf7ffd708   0x00b2fff4   0x00000000

(gdb) x/32x 0x8048680
0x8048680:   0x0804842e   0x08048456   0x08048438   0x08048442
0x8048690:   0x08048446   0x0804844c   0x776f7262   0x70000a6e
0x80486a0:   0x0a6b6e69   0x64657200   0x6877000a   0x0a657469
0x80486b0:   0x616c6200   0x000a6b63   0x61757161   0x6f67000a
0x80486c0:   0x000a646c   0x65756c62   0x636f000a   0x0a657268
0x80486d0:   0x75616d00   0x000a6576   0x6e617963   0x7267000a
0x80486e0:   0x000a7965   0x0a6e6174   0x69656200   0x000a6567
0x80486f0:   0x65657267   0x6f000a6e   0x0a78796e   0x73657400

(gdb) x/16x 0x80498e0
0x80498e0:   0x08048698   0x0804869f   0x080486a5   0x080486aa
0x80498f0:   0x080486b1   0x080486b8   0x080486be   0x080486c4
0x8049900:   0x080486ca   0x080486d1   0x080486d8   0x080486de
0x8049910:   0x080486e4   0x080486e9   0x080486f0   0x080486f7

(gdb) x/128x 0x8048690
0x8048690:   0x08048446   0x0804844c   0x776f7262   0x70000a6e
0x80486a0:   0x0a6b6e69   0x64657200   0x6877000a   0x0a657469
0x80486b0:   0x616c6200   0x000a6b63   0x61757161   0x6f67000a
0x80486c0:   0x000a646c   0x65756c62   0x636f000a   0x0a657268
0x80486d0:   0x75616d00   0x000a6576   0x6e617963   0x7267000a
0x80486e0:   0x000a7965   0x0a6e6174   0x69656200   0x000a6567
0x80486f0:   0x65657267   0x6f000a6e   0x0a78796e   0x73657400
0x8048700:   0x676e6974   0x64252020   0x0000000a   0x3b031b01
```

Problem 3: Omega Solution

Question 2 and 3: Value of %eax and %edx at 0x804846d before execution.



```
8048420:  83 f8 05              sub    $0x64,%eax
8048423:  77 31                 cmp    $0x5,%eax
8048425:  8b 04 85 80 86 04 08  ja     8048456 <func0+0x42>
804842c:  ff e0                 mov    0x8048680(,%eax,4),%eax
804842e:  83 45 08 01           jmp    *%eax
8048432:  83 45 0c 01           addl   $0x1,0x8(%ebp)
8048436:  eb 2c                 addl   $0x1,0xc(%ebp)
8048438:  83 45 08 02           jmp    8048464 <func0+0x50>
804843c:  83 45 0c 02           addl   $0x2,0x8(%ebp)
8048440:  eb 22                 addl   $0x2,0xc(%ebp)
8048442:  83 45 08 01           jmp    8048464 <func0+0x50>
8048446:  83 45 0c 02           addl   $0x1,0x8(%ebp)
804844a:  eb 18                 addl   $0x2,0xc(%ebp)
804844c:  83 45 08 02           jmp    8048464 <func0+0x50>
8048450:  83 45 0c 01           addl   $0x2,0x8(%ebp)
8048454:  eb 0e                 addl   $0x1,0xc(%ebp)
8048456:  c7 45 08 00 00 00 00  jmp    8048464 <func0+0x50>
804845d:  c7 45 0c 00 00 00 00  movl   $0x0,0x8(%ebp)
8048464:  8b 55 08              movl   $0x0,0xc(%ebp)
8048467:  8b 45 0c              mov    0x8(%ebp),%edx
804846a:  c1 e2 02              mov    0xc(%ebp),%eax
804846d:  8d 04 02              shl    $0x2,%edx
                                lea    (%edx,%eax,1),%eax
```

We don't jump
eax = eax*4 + 0x8048680
= 0x8048688

Jump to address 0x08048438
Add 2 to locations 0x8(ebp)
Add 2 to locations 0xc(ebp)
Jump to 0x8048464

edx = 3
eax = 3
edx = 0x3 << 2
edx = 0xc
eax = **0x3**, edx = **0xc**

Problem 3: Omega Solution

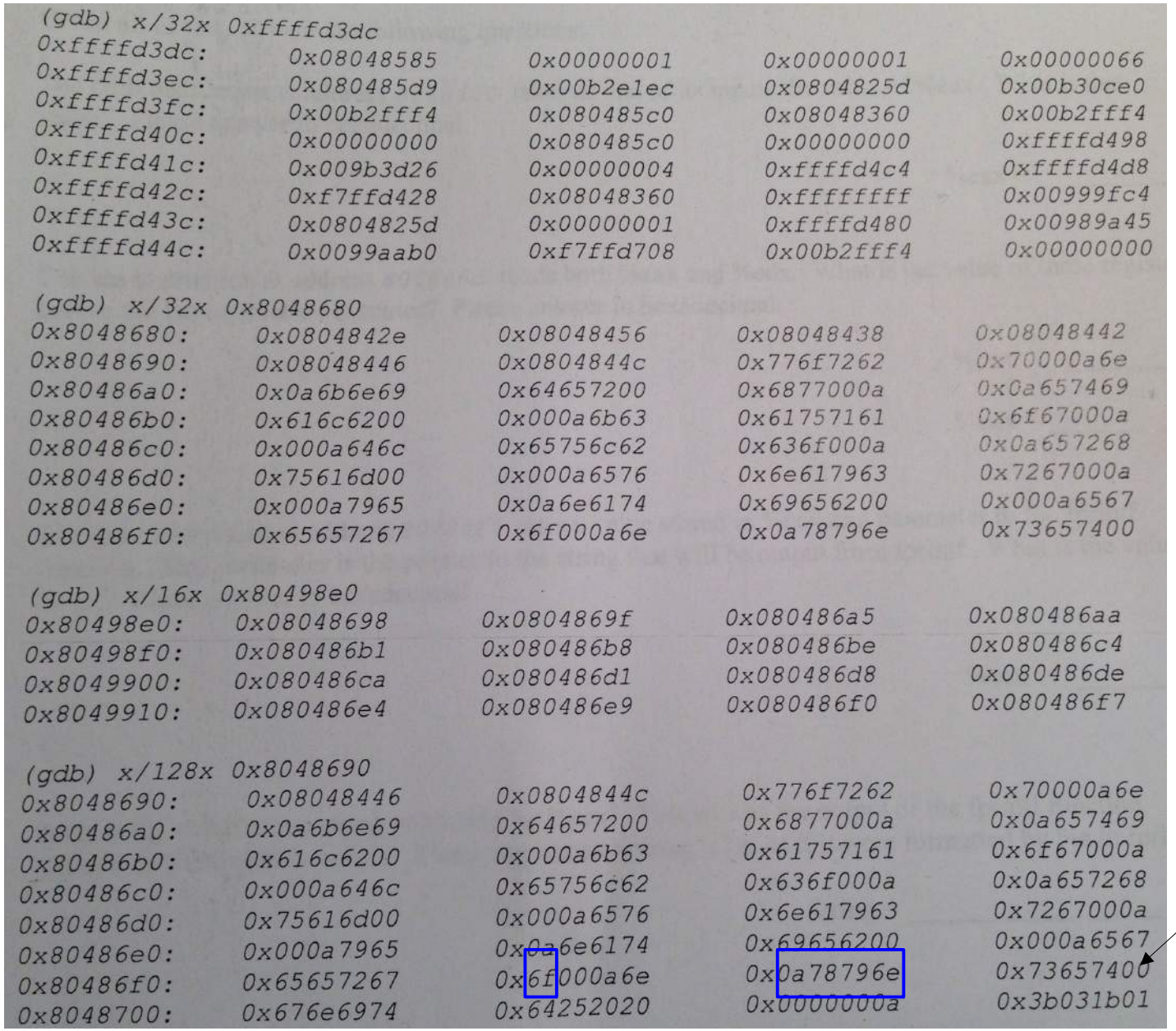
Question 4: value of ecx at instruction address 0x8048481

```
804846d:    8d 04 02                lea    (%edx,%eax,1),%eax
8048470:    8b 0c 85 e0 98 04 08    mov    0x80498e0(,%eax,4),%ecx
8048477:    ba 74 86 04 08          mov    $0x8048674,%edx
804847c:    a1 c0 98 04 08          mov    0x80498c0,%eax
8048481:    89 4c 24 08             mov    %ecx,0x8(%esp)
8048485:    89 54 24 04             mov    %edx,0x4(%esp)
8048489:    89 04 24                mov    %eax,(%esp)
804848c:    e8 b7 fe ff ff          call   8048348 <fprintf@plt>
8048491:    8b 45 10                mov    0x10(%ebp),%eax
8048494:    c9                      leave
8048495:    c3                      ret
```

eax = 0xf
ecx = *(0x80498e0
+ 0xf*4)
= *(0x804991C)
= **0x080486f7**

Problem 3: Omega Solution

Question 5: string output



```
(gdb) x/32x 0xffffd3dc
0xffffd3dc:   0x08048585   0x00000001   0x00000001   0x00000066
0xffffd3ec:   0x080485d9   0x00b2e1ec   0x0804825d   0x00b30ce0
0xffffd3fc:   0x00b2fff4   0x080485c0   0x08048360   0x00b2fff4
0xffffd40c:   0x00000000   0x080485c0   0x00000000   0xffffd498
0xffffd41c:   0x009b3d26   0x00000004   0xffffd4c4   0xffffd4d8
0xffffd42c:   0xf7ffd428   0x08048360   0xffffffff   0x00999fc4
0xffffd43c:   0x0804825d   0x00000001   0xffffd480   0x00989a45
0xffffd44c:   0x0099aab0   0xf7ffd708   0x00b2fff4   0x00000000

(gdb) x/32x 0x8048680
0x8048680:   0x0804842e   0x08048456   0x08048438   0x08048442
0x8048690:   0x08048446   0x0804844c   0x776f7262   0x70000a6e
0x80486a0:   0x0a6b6e69   0x64657200   0x6877000a   0x0a657469
0x80486b0:   0x616c6200   0x000a6b63   0x61757161   0x6f67000a
0x80486c0:   0x000a646c   0x65756c62   0x636f000a   0x0a657268
0x80486d0:   0x75616d00   0x000a6576   0x6e617963   0x7267000a
0x80486e0:   0x000a7965   0x0a6e6174   0x69656200   0x000a6567
0x80486f0:   0x65657267   0x6f000a6e   0x0a78796e   0x73657400

(gdb) x/16x 0x80498e0
0x80498e0:   0x08048698   0x0804869f   0x080486a5   0x080486aa
0x80498f0:   0x080486b1   0x080486b8   0x080486be   0x080486c4
0x8049900:   0x080486ca   0x080486d1   0x080486d8   0x080486de
0x8049910:   0x080486e4   0x080486e9   0x080486f0   0x080486f7

(gdb) x/128x 0x8048690
0x8048690:   0x08048446   0x0804844c   0x776f7262   0x70000a6e
0x80486a0:   0x0a6b6e69   0x64657200   0x6877000a   0x0a657469
0x80486b0:   0x616c6200   0x000a6b63   0x61757161   0x6f67000a
0x80486c0:   0x000a646c   0x65756c62   0x636f000a   0x0a657268
0x80486d0:   0x75616d00   0x000a6576   0x6e617963   0x7267000a
0x80486e0:   0x000a7965   0x0a6e6174   0x69656200   0x000a6567
0x80486f0:   0x65657267   0x6f000a6e   0x0a78796e   0x73657400
0x8048700:   0x676e6974   0x64252020   0x0000000a   0x3b031b01
```

0x6f = 'o'
0x6e = 'n'
0x79 = 'y'
0x78 = 'x'
0x1a = '\n'

NULL