

3

1	w	+	+	
2	b			+
3	l			
4	b			
5	l			
6	l			
7	w			
8	l	+	+	

4

1		0xFF		'\$'
2	16			'b'
3				
4	16		16	'w'
5	8			
6				
7	ESX			
8				

5

		3.12	5
src_type	dst_type		
char	int	movsbl %al, (%edx)	
int	char	movb %al, (%edx)	
int	unsigned	movl %eax, (%edx)	
short	int	movswl %ax, (%edx)	
unsigned char	unsigned	movzbl %al, (%edx)	
char	unsigned	movsbl %al, (%edx)	
int	int	movl %eax, (%edx)	

6

```
1  xptr  yptr  zptr                                R[ebp]+8  R[ebp]+12
   R[ebp]+16
2      func  C
void func(int *xptr, int *yptr, int *zptr)
{
    int tempx=*xptr;
    int tempy=*yptr;
    int tempz=*zptr;

    *yptr=tempx;
    *zptr = tempy;
    *xptr = tempz;
}
```

7

- 1 R[edx]=x
- 2 R[edx]=x+y+4
- 3 R[edx]=x+8\*y
- 4 R[edx]=y+2\*x+12
- 5 R[edx]=4\*y
- 6 R[edx]=x+y

8

1 R[edx] R[edx]+M[R[eax]]=0x00000080+M[0x8049300] EDX  
00000080H+FFFFFFFF0H

0000 0000 0000 0000 0000 0000 1000 0000  
+ 1111 1111 1111 1111 1111 1111 1111 0000  
-----  
1 0000 0000 0000 0000 0000 0000 0111 0000

EDX 0x00000070 3.5 OF SF ZF CF OF=0  
ZF=0 SF=0 CF=1

2 R[ecx] R[ecx]-M[R[eax]+R[ebx]]=0x00000010+M[0x8049400], ECX  
00000010H-80000008H

0000 0000 0000 0000 0000 0000 0001 0000  
+ 0111 1111 1111 1111 1111 1111 1111 1000  
-----  
0 1000 0000 0000 0000 0000 0000 1000

ECX 0x80000008 3.5 OF SF ZF CF OF=1  
ZF=0 SF=1 CF=1 O=1

3 R[bx] R[bx] or M[R[eax]+R[ecx]\*8+4] BX  
0x0100 or M[0x8049384]=0100H or FF00H

0000 0001 0000 0000  
or 1111 1111 0000 0000  
-----  
1111 1111 0000 0000

BX 0xFF00 3.3.3 OR OF=CF=0 0 ZF=0  
1 SF=1

4 test “ ” R[dl] and 0x80

1000 0000  
and 1000 0000  
-----  
1000 0000

3.3.3 TEST OF=CF=0 0 ZF=0 1 SF=1

5 M[R[eax]+R[edx]] M[R[eax]+R[edx]]\*32 0x8049380  
M[0x8049380]\*32=0x908f12a8\*32 0x908f12a8 5

1001 0000 1000 1111 0001 0010 1010 1000<<5  
=0001 0001 1110 0010 0101 0101 0000 0000  
0x8049380 0x11e25500  
3.5 OF=1

6 R[cx] R[cx]-1 CX

0000 0000 0001 0000  
+ 1111 1111 1111 1111  
-----  
1 0000 0000 0000 1111

CX 0x0010 0x000F 3.5 DEC OF ZF SF  
OF=0 ZF=0 SF=0

9

movl 12(%ebp), %ecx //R[ecx] M[R[ebp]+12] y ECX  
sall \$8, %ecx //R[ecx] R[ecx]<<8 y\*256 ECX  
movl 8(%ebp), %eax //R[eax] M[R[ebp]+8] x EAX  
movl 20(%ebp), %edx //R[edx] M[R[ebp]+20] k EDX  
imull %edx, %eax //R[eax] R[eax]\*R[edx] x\*k EAX  
movl 16(%ebp), %edx //R[edx] M[R[ebp]+16] z EDX  
andl \$65520, %edx //R[edx] R[edx] and 65520 z&0xFFFF0 EDX  
addl %ecx, %edx //R[edx] R[edx] + R[ecx] z&0xFFFF0+y\*256 EDX  
subl %edx, %eax //R[eax] R[eax]-R[edx] x\*k-(z&0xFFFF0+y\*256) EAX

3

3 int v = x\*k-(z&0xFFFF0+y\*256) ;

10

R[ebp]+16 2 4 y 8 R[ebp]+20 4 32 y<sub>h</sub> num\_type  
4 32 y<sub>l</sub> 4 y  
unsigned long long  
movl 12(%ebp), %eax //R[eax] M[R[ebp]+12] x EAX  
movl 20(%ebp), %ecx //R[ecx] M[R[ebp]+20] y<sub>h</sub> ECX  
imull %eax, %ecx //R[ecx] R[ecx]\*R[eax] y<sub>h</sub>\*x 32 ECX  
mull 16(%ebp) //R[edx]R[eax] M[R[ebp]+16]\*R[eax] y<sub>l</sub>\*x EDX-EAX  
leal (%ecx, %edx), %edx  
//R[edx] R[ecx]+R[edx] y<sub>l</sub>\*x 32 y<sub>h</sub>\*x 32 EDX  
movl 8(%ebp), %ecx //R[ecx] M[R[ebp]+8] d ECX  
movl %eax, (%ecx) //M[R[ecx]] R[eax] x\*y 32 d 32  
movl %edx, 4(%ecx) //M[R[ecx]+4] R[edx] x\*y 32 d 32

11

3.3.4

PC +  
1 je 01110100 7408H 08H  
0x804838c+2+0x8=0x8048396  
call 0x80483b1=0x804838e+5+0x1e call 4  
IA -32 0000001EH call 5  
0x804838e 5  
2 jb F6H 0x8048390+2+0xf6=0x8048488  
movl 10 8  
0804A800H 0x804a800 4 00000001H  
0x1

3	jle	7EH	16H	Ox80492e0
mov	x	x	Ox80492e0=x+Ox16	x=Ox80492e0- Ox16=Ox80492ca
4	jmp	E9H	4	FFFFFF00H
-100H=-256	sub	Ox804829b	jmp	
Ox804829b+Oxffffffff00=Ox804829b- Ox100=Ox804819b				

12

1

1	movb	8(%ebp), %dl	//R[dl] M[R[ebp]+8]	x	DL
2	movl	12(%ebp), %eax	//R[ecx] M[R[ebp]+12]	p	EAX
3	testl	%eax, %eax	//R[ecx] and R[ecx]	p	0
4	je	.L1	// p 0 .L1		
5	testb	\$0x80, %dl	//R[dl] and 80H	x	0
6	je	.L1	// x>=0 .L1		
7	addb	%dl, (%eax)	//M[R[ecx]] M[R[ecx]]+R[dl]	*p+=x	
8	.L1:				
	C	if			

EFLAGS

EFLAGS

2	3.22	" if () goto ..."	C
---	------	-------------------	---

```

1 void comp(char x, int *p)
2 {
3     if (p!=0)
4         if (x<0)
5             *p += x;
6 }
```

13

```

1 int func(int x, int y)
2 {
3     int z = x*y;
4     if ( x<=-100 ) {
5         if ( y>x )
6             z = x+y ;
7     } else
8         z = x-y ;
9     } else if ( x>=16 )
10         z = x & y ;
11     return z;
12 }
```

14

1	4	x	y	k	4
1	movw	8(%ebp), %bx	//R[bx] M[R[ebp]+8]	x	BX
2	movw	12(%ebp), %si	//R[si] M[R[ebp]+12]	y	SI
3	movw	16(%ebp), %cx	//R[cx] M[R[ebp]+16]	k	CX
4	.L1:				

5	movw	%si, %dx	//R[dx] R[si] y DX
6	movw	%dx, %ax	//R[ax] R[dx] y AX
7	sarw	\$15, %dx	//R[dx] R[dx]>>15 y 16 DX
8	idiv	%cx	//R[dx] R[dx-ax]÷ R[cx] y%k DX
			//R[ax] R[dx-ax]÷ R[cx] y/k AX
9	imulw	%dx, %bx	//R[bx] R[bx]*R[dx] x*(y%k) BX
10	decw	%cx	//R[cx] R[cx]-1 k-1 CX
11	testw	%cx, %cx	//R[cx] and R[cx] OF=CF=0 SF=1, ZF=1
12	jle	.L2	// k 0 .L2
13	cmpw	%cx, %si	//R[si] - R[cx] y k
14	jg	.L1	// y k .L1
15	.L2:		
16	movswl	%bx, %eax	//R[eax] R[bx] x*(y%k) AX
2			BX SI AX CX DX
			EBX ESI
3		8	32
16		16	DX 16 AX

15

```

1  int f1(unsigned x)
2  {
3      int y = 0;
4      while (x!=0) {
5          y ^= x;
6          x >>= 1;
7      }
8      return y & 0x1;
9  }

```

f1	( x ^ x>>1 ^ x>>2 ^ ....) & 0x1	f1	x	x	1
----	---------------------------------	----	---	---	---

1

16

sw	x	2~5	x+3>7	.L7
	x			

```

x+3=0 .L7
x+3=1 .L2
x+3=2 .L2
x+3=3 .L3
x+3=4 .L4
x+3=5 .L5
x+3=6 .L7
x+3=7 .L6

```

```

switch (x)
case -2:
case -1:
..... // .L2
break;
case 0:

```

```

..... //L 3
break;
case 1:
..... //L 4
break;
case 2:
..... //L 5
break;
case 4:
..... //L 6
break;
default:
..... //L 7

```

17

```

      2 3      a char      p      short      4 5
b c      unsigned short      6      test      unsigned int      test
unsigned int test(char a, unsigned short b, unsigned short c, short *p);

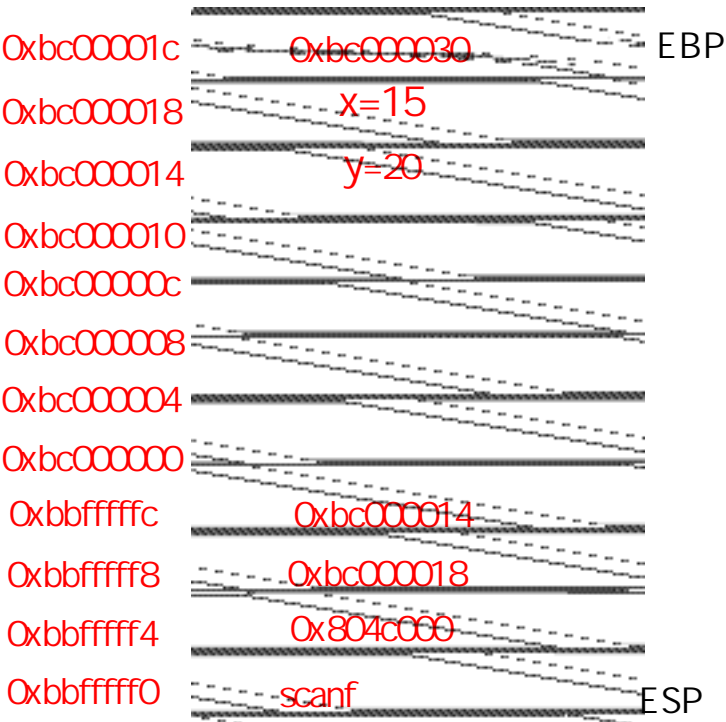
```

18

```

      pushl      R[esp]=R[esp]-4      2      R[esp]=0xbc00001c
1      3      R[ebp]=R[esp]=0xbc00001c      12      EBP
      10      EBP      0xbc00001c      13      EBP
      funct      0xbc000030
2      3      R[esp]=0xbc00001c      4      R[esp]= R[esp]-40=0xbc00001c-0x28=0xbbffffff4
      10      scanf      ESP      0xbbffffff4-4=0xbbffffff0      scanf
      ESP      0xbbffffff4      13      ESP      funct
      R[esp]=0xbc000020
3      5 6      scanf      &y      R[ebp]-8=0xbc000014      7 8
      scanf      &x      R[ebp]-4=0xbc000018      x y      0xbc000018
      0xbc000014
4      10      funct

```



19

```

      1      x      EBX      4      x=0      .L2      5~10
5 6 7      nx      x>>1      9 10 11      (x&1)+rv

```

C

```
1 int refunc(unsigned x) {
2     if ( x==0 )
3         return 0 ;
4     unsigned nx = x>>1 ;
5     int rv = refunc(nx) ;
6     return (x & 0x1) + rv
7 }
```

x                      1

20

IA-32	GCC	long double	12	10
		B	B	i
char A[10]	1	10	&A[0]	&A[0]+i
int B[100]	4	400	&B[0]	&B[0]+4i
short *C[5]	4	20	&C[0]	&C[0]+4i
short **D[6]	4	24	&D[0]	&D[0]+4i
long double E[10]	12	120	&E[0]	&E[0]+12i
long double *F[10]	4	40	&F[0]	&F[0]+4i

21

S	short *	As	leal (%edx), %eax
S+i	short *	As+2*i	leal (%edx, %ecx, 2), %eax
S[i]	short	M[As+2*i]	movw (%edx, %ecx, 2), %ax
&S[10]	short *	As+20	leal 20(%edx), %eax
&S[i+2]	short *	As+2*i+4	leal 4(%edx, %ecx, 2), %eax
&S[i]-S	int	(As+2*i-As)/2=i	movl %ecx, %eax
S[4*i+4]	short	M[As+2*(4*i+4)]	movw 8(%edx, %ecx, 8), %ax
*(S+i-2)	short	M[As+2*(i-2)]	movw -4(%edx, %ecx, 2), %ax

22

	EAX				
M[a+28*i+4*j]+M[b+20*j+4*i]	a	b	int	4B	M=5, N=7

23

11	a[i][j][k]	a+4*(63*i+9*j+k)	M=9	N=63/9=7	12
a	4536	L=4536/(4*L*M)=18			

24

1	M=76/4=19	EDI	j	ECX
2		trans_matrix	C	
1	void trans_matrix(int a[M][M]) {			
2	int i, j, t *p;			
3	int c=(M<<2);			
3	for (i = 0; i < M; i++) {			
4	p=&a[0][i];			
5	for (j = 0; j < M; j++) {			

```
6      t=*p;
7      *p = a[i][j];
8      a[i][j] = t
9      p += c
10     }
11 }
12 }
```

25

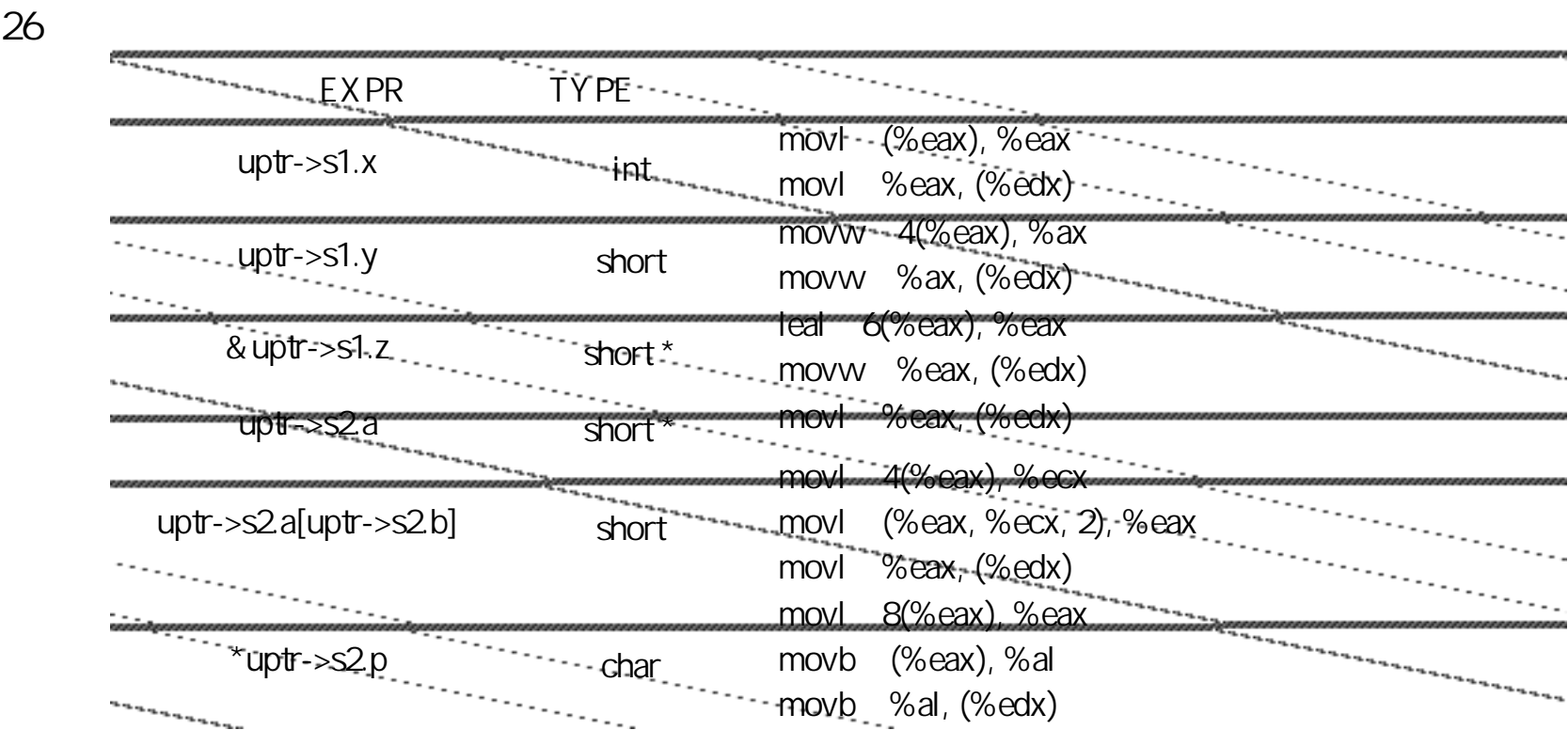
1 node

4+(4+4)+4=16

p sx sy next

0 4 8 12

```
2 np_init
void np_init(struct node *np)
{
    np->s.x = np->s.y ;
    np->p = &(np->s.x) ;
    np->next= np ;
}
```



27

1	S1:	s	c	i	d								
		0	2	4	8	12	4						
2	S2:	i	s	c	d								
		0	4	6	7	8	4						
3	S3:	c	s	i	d								
		0	2	4	8	12	4						
4	S4:	s	c										
		0	6		8	2							
5	S5:	c	s	i	d	e							
		0	4	8	12	16	24	4					
6	S6:	c	s	d									
		0	36	40		44	4						

Linux double 4

28

Windows

c	d	i	s	p	l	g	v
0	8	16	20	24	28	32	40

48

d g 8

4

44+4=48

struct {

double d;

long long g

int i;

char \*p;

long l;

void \*v

short s;

char c;

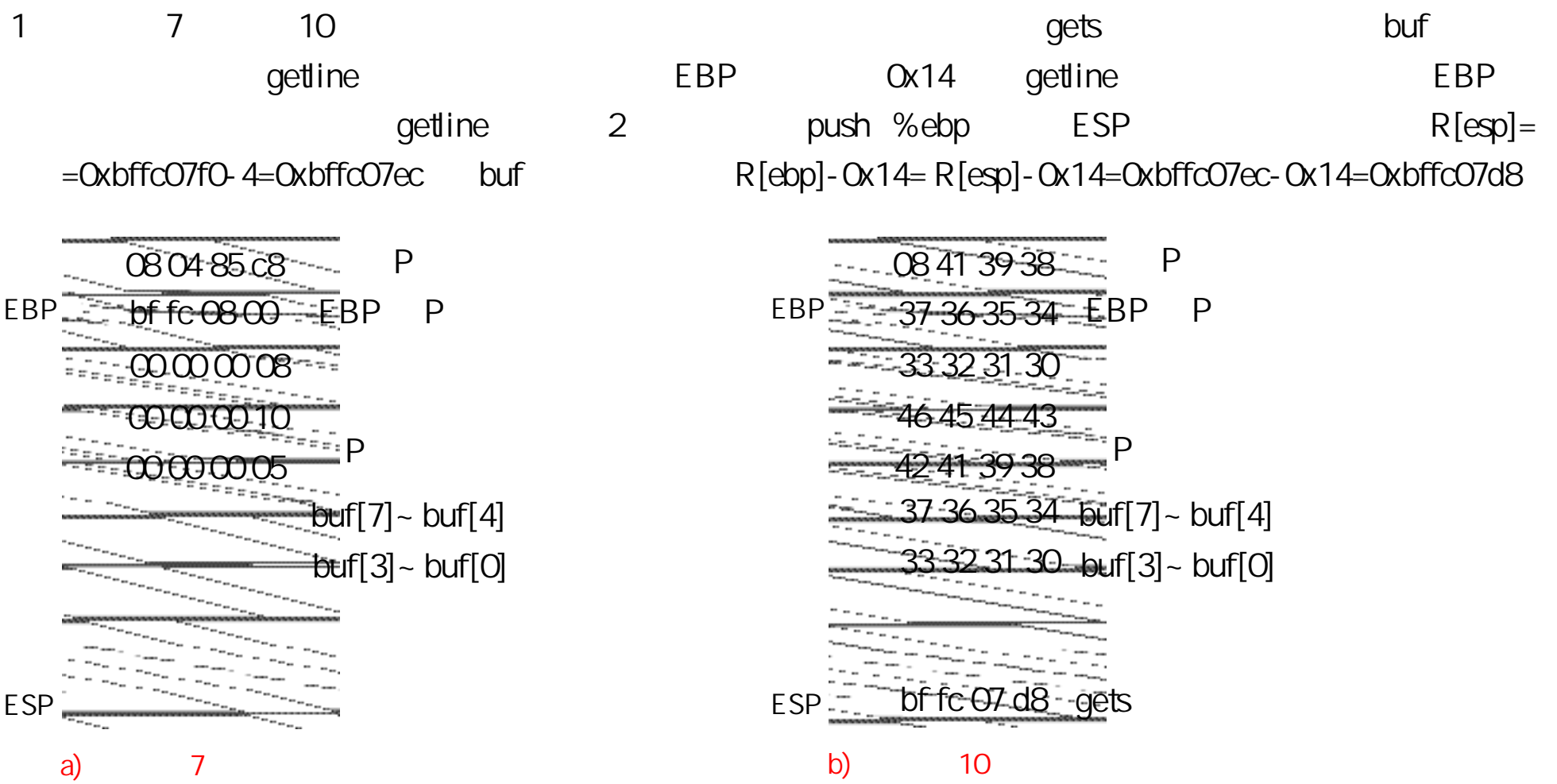
} test

d g i p l v s c

0 8 16 20 24 28 32 34

34+6=40

29



```
viod abc(long c, long *a, int *b);  
viod abc(unsigned long c, long *a, int *b);
```

31

5 C

32

11 12 x 0xc8-4=200-4=196B LEN=196/28=7

33

```
1 n1.ptr n1.data1 n2.data2 n2next 0 4 0 4
2 node 8
3 chain_proc C
  uptr->n2next->n1.data1 = *(uptr->n2_next->n1.ptr) - uptr->n2_data2;
```

34

```
1 trace tptr RDI
2 trace C
long trace( tree_ptr tptr)
{
    long ret_val=0;
    tree_ptr p=tptr;
    while (p!=0) {
        ret_val=p->val;
        p=p->left;
    }
    return ret_val;
}
3 trace val
```