Лекция 8

9 марта

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                        // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
        push    ebp                ;  (1)
        mov     ebp, esp           ;  (2)
        push    esi                ;  (3)
        push    ebx                ;  (4)
        mov     esi, DWORD [ebp+8] ;  (5)
        mov     eax, DWORD [esi+4] ;  (6)
        lea     edx, [eax+1]       ;  (7)
        and     eax, -2147483648   ;  (8)
        and     edx,  2147483647   ;  (9)
        or      eax, edx           ; (10)
        mov     DWORD [esi+4], eax ; (11)
        movzx   ebx, BYTE [esi+1]  ; (12)
        sal     ebx, 6             ; (13)
        sar     bl, 6              ; (14)
        lea     ecx, [0+ebx*8]     ; (15)
        movzx   edx, BYTE [esi]    ; (16)
        mov     eax, edx           ; (17)
        and     edx, 7             ; (18)
        sal     eax, 5             ; (19)
        sar     al, 5              ; (20)
        or      ecx, eax           ; (21)
        sal     ecx, 3             ; (22)
        sar     cl, 3              ; (23)
        lea     eax, [0+ecx*8]     ; (24)
        or      edx, eax           ; (25)
        mov     BYTE [esi], dl     ; (26)
        sal     edx, 5             ; (27)
        sar     dl, 5              ; (28)
        add     ebx, edx           ; (29)
        add     ebx, ecx           ; (30)
        mov     BYTE [esi+8], bl   ; (31)
        pop     ebx                ; (32)
        pop     esi                ; (33)
        pop     ebp                ; (34)
        ret                        ; (35)
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                    // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     mov     esi, DWORD [ebp+8] ;  (5)
     mov     eax, DWORD [esi+4] ;  (6)
     lea     edx, [eax+1]       ;  (7)
     and     eax, -2147483648   ;  (8)
     and     edx,  2147483647   ;  (9)
     or      eax, edx           ; (10)
     mov     DWORD [esi+4], eax ; (11)
     ; пропуск
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                          // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     movzx   ebx, BYTE [esi+1]  ; (12)
     sal     ebx, 6             ; (13)
     sar     bl, 6              ; (14)
     lea     ecx, [0+ebx*8]     ; (15)
     movzx   edx, BYTE [esi]    ; (16)
     mov     eax, edx           ; (17)
     and     edx, 7             ; (18)
     sal     eax, 5             ; (19)
     sar     al, 5              ; (20)
     or      ecx, eax           ; (21)
     sal     ecx, 3             ; (22)
     sar     cl, 3              ; (23)
     lea     eax, [0+ecx*8]     ; (24)
     or      edx, eax           ; (25)
     mov     BYTE [esi], dl     ; (26)
     ; пропуск
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                      // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     sal     edx, 5              ; (27)
     sar     dl, 5               ; (28)
     add     ebx, edx            ; (29)
     add     ebx, ecx            ; (30)
     mov     BYTE [esi+8], bl    ; (31)
     pop     ebx                 ; (32)
     pop     esi                 ; (33)
     pop     ebp                 ; (34)
     ret                         ; (35)
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                          // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     movzx   ebx, BYTE [esi+1]  ; p->c
     sal     ebx, 6             ; <<
     sar     bl, 3              ; 3
     movzx   edx, BYTE [esi]    ;
     mov     eax, edx           ;
     and     edx, 7             ;
     sal     eax, 5             ;
     sar     al, 5              ; p->a
     or      ebx, eax           ;
     sal     ebx, 3             ;
     or      edx, ebx           ;
     mov     BYTE [esi], dl     ;
     ; пропуск
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                          // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     movzx   ebx, BYTE [esi+1]  ; p->c
     sal     ebx, 6             ; <<
     sar     bl, 3              ; 3
     movzx   edx, BYTE [esi]    ;
     mov     eax, edx           ;
     and     edx, 7             ;
     sal     eax, 5             ;
     sar     al, 5              ; p->a
     or      ebx, eax           ;
     sal     ebx, 3             ;
     or      edx, ebx           ;
     mov     BYTE [esi], dl     ;
     ; пропуск
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                          // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     movzx   ebx, BYTE [esi+1]  ; p->c
     sal     ebx, 6             ; <<
     sar     bl, 3              ; 3
     movzx   edx, BYTE [esi]    ;
     mov     eax, edx           ;
     and     edx, 7             ;
     sal     eax, 5             ;
     sar     al, 5              ; p->a
     or      ebx, eax           ;
     sal     ebx, 3             ;
     or      edx, ebx           ;
     mov     BYTE [esi], dl     ;
     ; пропуск
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                        // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     movzx   ebx, BYTE [esi+1]  ; p->c
     sal     ebx, 6             ; <<
     sar     bl, 3              ; 3
     movzx   edx, BYTE [esi]    ;
     mov     eax, edx           ;
     and     edx, 7             ;
     sal     eax, 5             ;
     sar     al, 5              ; p->a
     or      ebx, eax           ;
     sal     ebx, 3             ;
     or      edx, ebx           ;
     mov     BYTE [esi], dl     ;
     ; пропуск
```

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;
   unsigned cntr: 31;
   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                        // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}
```

```
section .text
global f
f:
     ; пропуск
     movzx   ebx, BYTE [esi+1]  ;
     sal     ebx, 6             ;
     sar     bl, 6              ; p->c
     movzx   edx, BYTE [esi]    ;
     sal     edx, 5             ;
     sar     dl, 5              ; p->a
     movzx   ecx, BYTE [esi]    ;
     sar     cl, 3              ; p->b
     add     ebx, edx           ;
     add     ebx, ecx           ;
     mov     BYTE [esi+8], bl   ;
     pop     ebx                ;
     pop     esi                ;
     pop     ebp                ;
     ret                        ;
```

Jcc	Условие	Описание
JE	ZF	Равно / Ноль
JNE	~ZF	Не равно / Не ноль
JS	SF	Отрицательное число
JNS	~SF	Неотрицательное число
JG	~(SF^OF)&~ZF	Больше (знаковые числа)
JGE	~(SF^OF)	Больше либо равно (знаковые числа)
JL	(SF^OF)	Меньше (знаковые числа)
JLE	(SF^OF)|ZF	Меньше либо равно (знаковые числа)
JA	~CF&~ZF	Больше (числа без знака)
JB	CF	Меньше (числа без знака)

```
int absdiff(int x, int y) {
    int result;
    if (x > y) {
      result = x-y;
    } else {
      result = y-x;
    }
    return result;
}
```

```
absdiff:
   push  ebp
   mov  ebp, esp
   mov  edx, dword [8 + ebp]  ; (1)
   mov  eax, dword [12 + ebp] ; (2)
   cmp  edx, eax              ; (3)
   jle  .L6                   ; (4)
   sub  edx, eax              ; (5)
   mov  eax, edx              ; (6)
   jmp  .L7                   ; (7)
.L6:                          ; (8)
   sub eax, edx               ; (9)
.L7:                          ; (10)
   pop ebp
   ret
```

```
int goto_ad(int x, int y) {
  int result;
  if (x <= y) goto Else;
  result = x-y;
  goto Exit;
Else:
  result = y-x;
Exit:
  return result;
}
```

```
absdiff:
   push  ebp
   mov   ebp, esp
   mov   edx, dword [8 + ebp]  ; (1)
   mov   eax, dword [12 + ebp] ; (2)
   cmp   edx, eax              ; (3)
   jle   .L6                   ; (4)
   sub   edx, eax              ; (5)
   mov   eax, edx              ; (6)
   jmp   .L7                   ; (7)
.L6:                           ; (8)
   sub eax, edx                ; (9)
.L7:                           ; (10)
   pop ebp
   ret
```

```
val = Test ? Then_Expr : Else_Expr;

val = x>y ? x-y : y-x;
```

```
  nt = !Test;
  if (nt) goto Else;
  val = Then_Expr;
  goto Done;
Else:
  val = Else_Expr;
Done:
  . . .
```

```
  tmp_val = Then_Expr;
  result = Else_Expr;
  t = Test;
  if (t) result = tmp_val;
  return result;
```

```
int absdiff(int x, int y) {
    int result;
    if (x > y) {
      result = x-y;
    } else {
      result = y-x;
    }
    return result;
}
```

---

x загружен в edi

y загружен в esi

```
absdiff:
        mov     edx, edi
        sub     edx, esi        # tmp_val:edx = x-y
        mov     eax, esi
        sub     eax, edi        # result:eax = y-x
        cmp     edi, esi        # Compare x:y
        cmovg   eax, edx        # If >, result:eax = tmp_val:edx
        ret
```

```
int pcount_do(unsigned x) {
  int result = 0;
  do {
    result += x & 0x1;
    x >>= 1;
  } while (x);
  return result;
}
```

```
int pcount_do(unsigned x)
{
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

```
int pcount_do(unsigned x)
{
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

```
   mov ecx, 0    ;   result = 0
.L2:             ; loop:
   mov eax, edx
   and eax, 1    ;   t = x & 1
   add ecx, eax  ;   result += t
   shr edx, 1    ;   x >>= 1
   jne .L2       ;   If !0, goto loop
```

- Распределение регистров:

edx x

ecx result

```
int pcount_while(unsigned x) {
  int result = 0;
  while (x) {
    result += x & 0x1;
    x >>= 1;
  }
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
  if (!x) goto done;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
done:
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  if (!x) goto done;
  result += x & 0x1;
  x >>= 1;
  goto loop;
done:
  return result;
}
```

```
#define WSIZE 8*sizeof(int)

int pcount_for(unsigned x) {
  int i;
  int result = 0;
  for (i = 0; i < WSIZE; i++) {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  return result;
}
```

```
#define WSIZE 8*sizeof(int)

int pcount_for(unsigned x) {
  int i;
  int result = 0;
  for (i = 0; i < WSIZE; i++) {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  return result;
}
```

```
int pcount_for_gt(unsigned x) {
  int i;
  int result = 0;
  i = 0;
  if (!(i < WSIZE))
    goto done;
 loop:
  {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  i++;
  if (i < WSIZE)
    goto loop;
 done:
  return result;
}
```

```
#define WSIZE 8*sizeof(int)

int pcount_for(unsigned x) {
  int i;
  int result = 0;
  for (i = 0; i < WSIZE; i++) {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  return result;
}
```

```
int pcount_for_gt(unsigned x) {
  int i;
  int result = 0;
  i = 0;
  if (!(i < WSIZE))
    goto done;
 loop:
  {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  i++;
  if (i < WSIZE)
    goto loop;
 done:
  return result;
}
```

```
int fib(unsigned x) {
  int i, res;
  int predpred = 0;
  int pred = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = predpred + pred;
     predpred = pred;
     pred = res;
  }
  return res;
}
```

```
int fib(unsigned x) {
  int i, res;
  int predpred = 0;
  int pred = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = predpred + pred;
     predpred = pred;
     pred = res;
  }
  return res;
}
```

```
int fib(int x) { // x >= 1
  int i;
  int predpred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = predpred + pred;
     predpred = pred;
     pred = res;
  }
  return res;
}
```

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; predpred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```