© 2023 МГУ/ВМК/СП

Лекция 6

1 марта

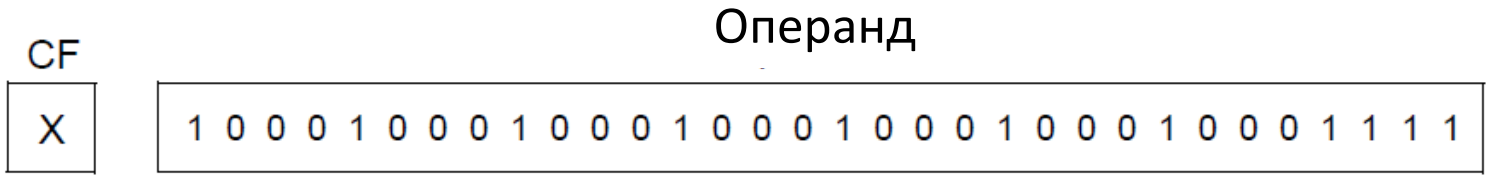
© 2023 МГУ/ВМК/СП

Сдвиги

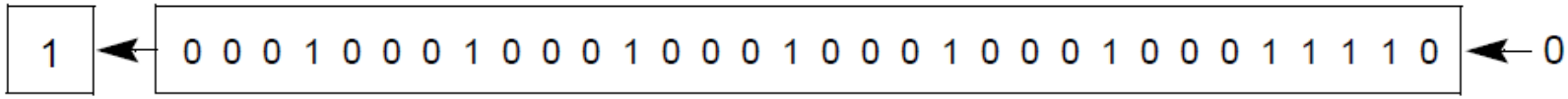
Данные	Направление сдвига: Влево	Направление сдвига: Вправо	
Беззнаковые	SHL	SHR	r/m 8/16/32, i8 r/m 8/16/32, cl
Знаковые	SAL	SAR	

Сдвиг влево

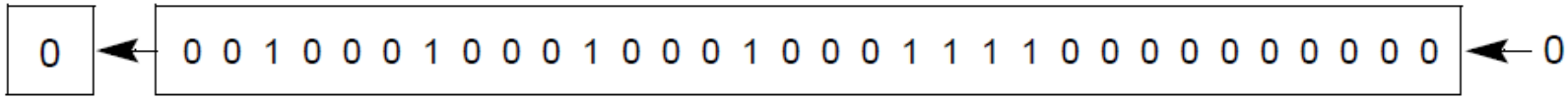
Начальное состояние



После сдвига на один разряд: SHL/SAL



После сдвига на 10 разрядов: SHL/SAL



© 2023 МГУ/ВМК/СП

Логический сдвиг вправо

Начальное состояние

Операнд

CF

1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 | X

После сдвига на один разряд: SHR

0 → 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 → 1

После сдвига на 10 разрядов: SHR

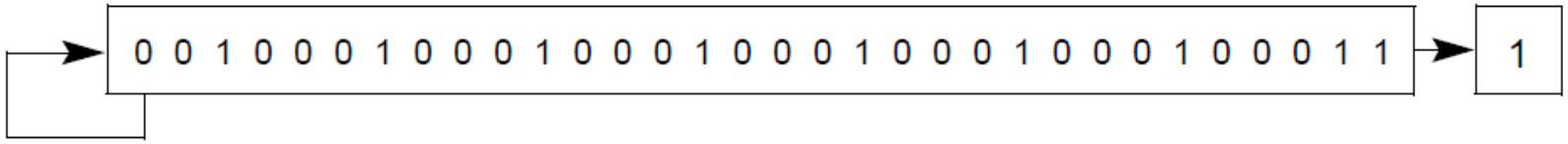
0 → 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 → 0

© 2023 МГУ/ВМК/СП

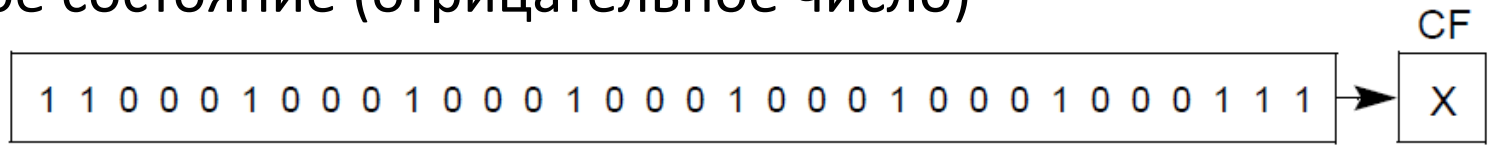
Арифметический сдвиг вправо

Начальное состояние (положительное число)

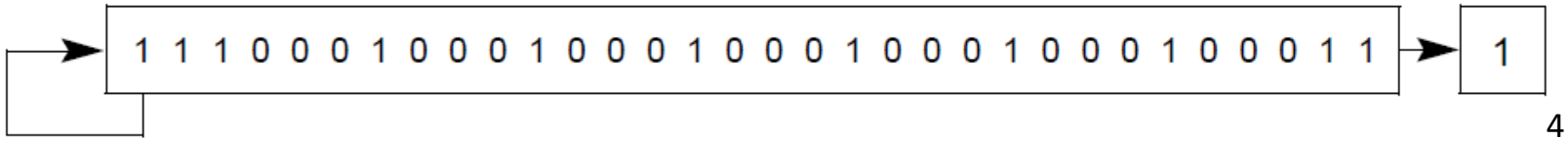
После сдвига на один разряд: SAR



Начальное состояние (отрицательное число)



После сдвига на один разряд: SAR



© 2023 МГУ/ВМК/СП

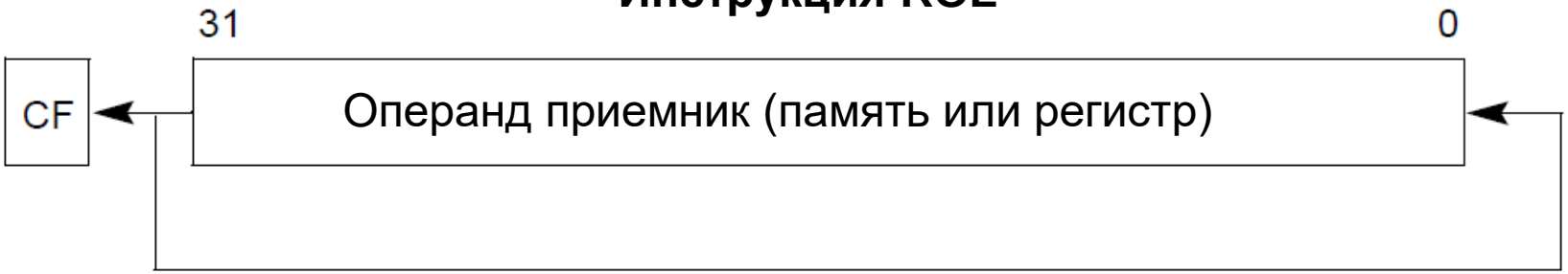
Пример

```
char upndown(char x) {
   return (x << 8) >> 8;
}
```

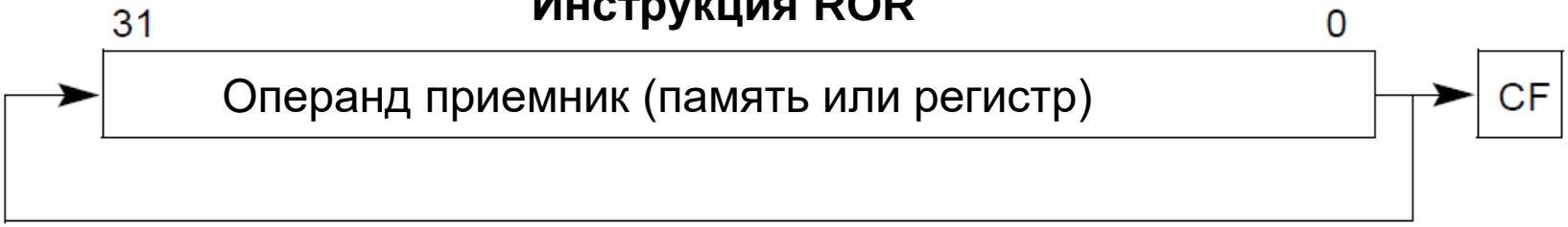
```
section .text
global upndown
upndown:
   push    ebp
   mov     ebp, esp
   movsx   eax, byte [ebp+8]
   sal     eax, 8
   sar     eax, 8
   pop     ebp
   ret
```

© 2023 МГУ/ВМК/СП

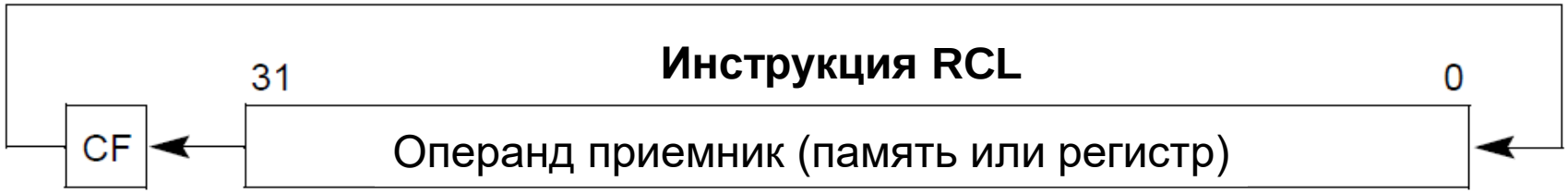
Инструкция ROL



Инструкция ROR



Инструкция RCL



Инструкция RCR

© 2023 МГУ/ВМК/СП

Пример: фрагмент кода криптографической хеш-функции

В 2013 году Intel были заявлены команды, вычисляющие криптографические хэш-функции:

- `SHA1RNDS4, SHA1NEXTE, SHA1MSG1, SHA1MSG2`
- `SHA256RNDS2, SHA256MSG1, SHA256MSG2`

```
unsigned sha256_f1(unsigned x) {
   unsigned t;
   t  = ((x >> 2)  | (x << ((sizeof(x) << 3) - 2)));  // (1)
   t ^= ((x >> 13) | (x << ((sizeof(x) << 3) - 13))); // (2)
   t ^= ((x >> 22) | (x << ((sizeof(x) << 3) - 22))); // (3)
   return t;
}
```

```
global sha256_f1
sha256_f1:
   push    ebp
   mov     ebp, esp
   mov     edx, dword [ebp+8] ; (1)
   pop     ebp                ; (2)
   mov     eax, edx           ; (3)
   mov     ecx, edx           ; (4)
   ror     eax, 13            ; (5)
   ror     ecx, 2             ; (6)
   xor     eax, ecx           ; (7)
   ror     edx, 22            ; (8)
   xor     eax, edx           ; (9)
   ret
```

Криптографические хеш-функции H(X):

- Необратимость
  H(X) = m, m – задано
- Стойкость к коллизиям первого рода
  H(M) = H(N), M – задано
- Стойкость к коллизиям второго рода
  $H(M_1) = H(M_2)$

© 2023 МГУ/ВМК/СП

Обратная задача

```
static int a, b, c, d;

???
```

```
mov     eax, dword [b] ; (1)
mov     edx, dword [c] ; (2)
or      al, -1         ; (3)
sal     eax, 3         ; (4)
add     edx, eax       ; (5)
mov     dword [a], eax ; (6)
mov     eax, edx       ; (7)
sar     edx, 31        ; (8)
idiv    dword [d]      ; (9)
mov     dword [c], eax ; (10)
```

© 2023 МГУ/ВМК/СП

«Основные» команды IA-32

Пересылка данных	Двоичная арифметика ✓	Передача управления	Логические ✓	Сдвиги и вращения	Битовые и байтовые	Прочее
• MOV	• ADD	• JMP	• AND	• SAR	• **SETcc**	• LEA
• XCHG	• **ADC**	• Jcc	• OR	• SHR	• TEST	• NOP
• BSWAP	• SUB	• CALL	• XOR	• SAL, SHL		
• MOVSX	• **SBB**	• RET	• NOT	• ROR		
• MOVZX	• NEG			• ROL		
• CDQ	• IMUL			• RCR		
• CWD	• MUL			• RCL		
• CBW	• IDIV					
• PUSH	• DIV					
• POP	• INC					
• **CMOVcc**	• DEC					
	• CMP					

Красным выделены не упоминавшиеся ранее на лекциях ассемблерные инструкции.
Команды двоичной арифметики и логические команды представлены в полном составе.

© 2023 МГУ/ВМК/СП

Сложение 64 разрядных чисел на 32 разрядных регистрах

```
long long f1(long long a, long long b) {
   long long c;
   c = a + b;
   return c;
}
```

```
; начало функции пропущено
   mov     eax, dword [ebp+16] ; (1)
   mov     edx, dword [ebp+20] ; (2)
   add     eax, dword [ebp+8]  ; (3)
   adc     edx, dword [ebp+12] ; (4)
; конец функции пропущен
```

© 2023 МГУ/ВМК/СП

Вычитание 64 разрядных чисел на 32 разрядных регистрах

```
long long f1(long long a, long long b) {
   long long c;
   c = a - b;
   return c;
}
```

```
; начало функции пропущено
   mov     eax, dword [ebp+8]  ; (1)
   mov     edx, dword [ebp+12] ; (2)
   sub     eax, dword [ebp+16] ; (3)
   sbb     edx, dword [ebp+20] ; (4)
; конец функции пропущен
```

© 2023 МГУ/ВМК/СП

Умножение 64 разрядных чисел на 32 разрядных регистрах

```
long long f2(long long a,
             long long b) {
   long long c;
   c = a * b;
   return c;
}
```

```
global f2
f2:
   push    ebp
   mov     ebp, esp
   push    ebx
   mov     eax, dword [ebp+8]
   mov     edx, dword [ebp+16]
   mov     ecx, dword [ebp+20]
   mov     ebx, dword [ebp+12]
   imul    ecx, eax
   imul    ebx, edx
   mul     edx
   add     ecx, ebx
   add     edx, ecx
   pop     ebx
   pop     ebp
   ret
```

© 2023 МГУ/ВМК/СП

Обоснование умножения на «коротких» регистрах (1/3)

$$\mathrm{U}(\vec{x}_w) = \sum_{i=0}^{w-1} 2^i x_i$$

Беззнаковая интерпретация w-разрядного битового вектора $\vec{x}_w$

$$T(\vec{x}_w) = -2^{w-1} x_{w-1} + \sum_{i=0}^{w-2} 2^i x_i$$

Знаковая интерпретация w-разрядного битового вектора $\vec{x}_w$

$$T(\vec{x}_{2w}) = -2^{2w-1} x_{2w-1} + \sum_{i=0}^{2w-2} 2^i x_i =$$

$$= 2^w \left(-2^{w-1} x_{2w-1} + \sum_{i=w}^{2w-2} 2^i x_{i+w} \right) + \sum_{i=0}^{w-1} 2^i x_i =$$

$$= 2^w T\left(\vec{x}_w^h\right) + \mathrm{U}\left(\vec{x}_w^l\right)$$

$\vec{x}_w^h$ – старшие w разрядов битового вектора $\vec{x}_{2w}$
$\vec{x}_w^l$ – младшие w разрядов

© 2023 МГУ/ВМК/СП

Обоснование умножения на «коротких» регистрах (2/3)

Обозначим

$$\mathrm{U}(\vec{x}_w) = \acute{x} \qquad T(\vec{x}_w) = x$$

Лемма

$$(\acute{x} * \acute{y}) mod 2^w = (\acute{x} * y) mod 2^w = (x * \acute{y}) mod 2^w = (x * y) mod 2^w$$

$$\acute{x} = x + 2^w x_{w-1}$$

© 2023 МГУ/ВМК/СП

Обоснование умножения на «коротких» регистрах (3/3)

$*_{2w}$ – умножение битовых векторов размера $2w$ с сохранением результата в виде вектора размера $2w$

$$\vec{y}_w^h : \vec{y}_w^l *_{2w} \vec{x}_w^h : \vec{x}_w^l =$$

$$= \left\{ 2^{2w} * T\left(\vec{y}_w^h\right) * T\left(\vec{x}_w^h\right) + 2^w * \left[T\left(\vec{y}_w^h\right) * U\left(\vec{x}_w^l\right) + U\left(\vec{y}_w^l\right) * T\left(\vec{x}_w^h\right) \right] + U\left(\vec{y}_w^l\right) * U\left(\vec{x}_w^l\right) \right\} mod 2^{2w}$$

разряды

4w	3w	2w	w	0	
					h*h
					h*l
					l*h
					l*l

h – старшая половина разрядов множителя, ***l*** – младшая.

© 2023 МГУ/ВМК/СП

Далее…

- Арифметика
- Операции над машинными словами
- Реализация управляющих операторов языка Си
  - goto-форма
  - Условная передача данных
  - Организация циклов
  - Оператор switch

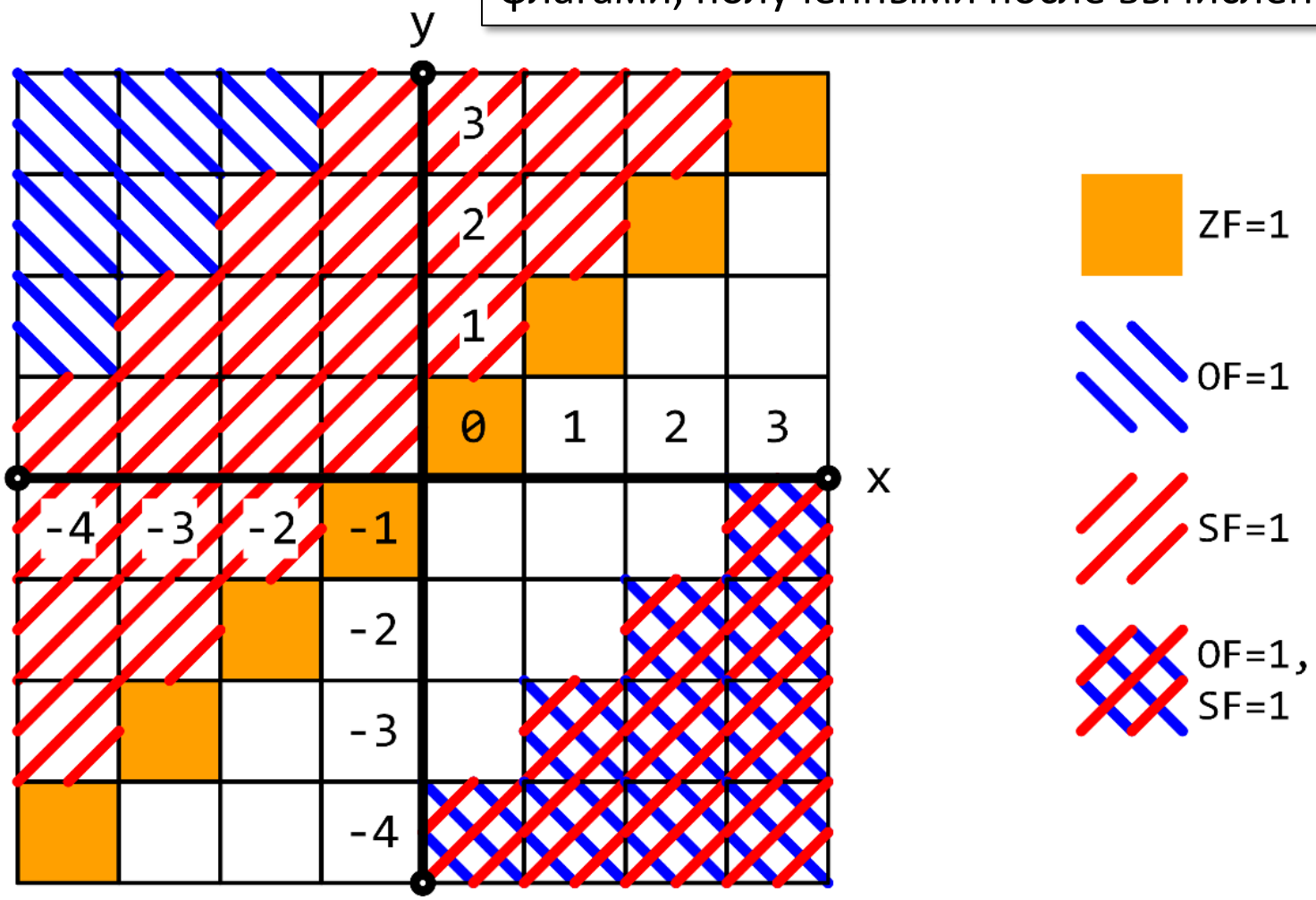
© 2023 МГУ/ВМК/СП

J*cc*	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)

© 2023 МГУ/ВМК/СП

Сравнение знаковых чисел

Сравнение x и y реализуется как формула над флагами, полученными после вычисления (x-y)



© 2023 МГУ/ВМК/СП

«Основные» команды IA-32

Пересылка данных	Двоичная арифметика ✓	Передача управления	Логические ✓	Сдвиги и вращения	Битовые и байтовые	Прочее
• MOV • XCHG • BSWAP • MOVSX • MOVZX • CDQ • CWD • CBW • PUSH • POP • **CMOVcc**	• ADD • ADC • SUB • SBB • NEG • IMUL • MUL • IDIV • DIV • INC • DEC • CMP	• JMP • J*cc* • CALL • RET	• AND • OR • XOR • NOT	• SAR • SHR • SAL, SHL • ROR • ROL • RCR • RCL	• **SET*cc*** • TEST	• LEA • NOP

Красным выделены не упоминавшиеся ранее на лекциях ассемблерные инструкции. Команды двоичной арифметики и логические команды представлены в полном составе.

© 2023 МГУ/ВМК/СП

Логический тип данных

- Начиная с C99 определен тип _Bool
  - Принимаемые значения `0` или `1`
  - Для хранения выделяется 1 байт
- `setcc r/m 8`

```
if (condition code) {
   ОП ← 1
} else {
   ОП ← 0
}
```

Задача на восстановление типов и операции сравнения

```
cmp_f:
   mov  eax, [esp+8]
   cmp  [esp+4], eax
   setl al
   ret
```

```
_Bool cmp_f(_____ x, _____ y) {
   return x _____ y;
}
```

© 2023 МГУ/ВМК/СП

GOTO-форма программы (1/3)

```c
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
```

© 2023 МГУ/ВМК/СП

GOTO-форма программы (2/3)

```
jmp label
```

~

```
goto label;
```

```
cmp …
jcc label
```

~

```
if (cc)
   goto label;
```

```
if (expr) {
   then_stmt;
} else {
   else_stmt;
}
```

```
_Bool t = !(expr);
if (t)
   goto else_label;
then_stmt;
goto end_label;
else_label:
else_stmt;
end_label:
```

```
   cmp …
   jcc .else
   then_stmt
   jmp .end
.else:
   else_stmt
.end
```

© 2023 МГУ/ВМК/СП

GOTO-форма программы (3/3)

```
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
```