© 2018 МГУ/ВМК/СП

Лекция 6

28 февраля

© 2018 МГУ/ВМК/СП

Дальнейший материал

- Взаимосвязь языка Си, языка ассемблера и особенностей архитектуры IA32
  - Операции над целыми числами и битовыми векторами
    - «быстрая» арифметика и обработка 64 разрядных чисел
    - побитовые операции, сдвиги, вращения
  - реализация управляющих конструкций языка Си
  - адресная арифметика
  - массивы
  - структуры и объединения, выравнивание данных
  - соглашение вызова
    - cdecl, stdcall, fastcall
    - выравнивание стека
    - ускорение вызова функций
    - переменное число параметров
  - числа с плавающей точкой
  - …

© 2018 МГУ/ВМК/СП

Группы команд

- *Общего назначения*
- *x87 FPU*
- MMX
- SSE
- IA-32e команды 64-разрядного режима работы
- Системные команды
- Аппаратная виртуализация

- *Передача данных*
- **Команды двоичной арифметики**
- Команды двоично-десятичной арифметики
- **Логические команды**
- **Сдвиги и вращения**
- **Битовые и байтовые команды**
- *Передача управления*
- **Строковые**
- *Ввод/Вывод*
- *Явное управление EFLAGS*
- *Вспомогательные*

© 2018 МГУ/ВМК/СП

Рассмотренные команды

- MOV
- **XCHG**
- **BSWAP**

- ADD
- SUB
- NEG
- IMUL
- MUL
- IDIV
- DIV
- INC
- DEC

- MOVSX
- MOVZX
- CDQ
- CWD
- CBW

- JMP
- Jcc
- CALL
- RET
- PUSH
- POP

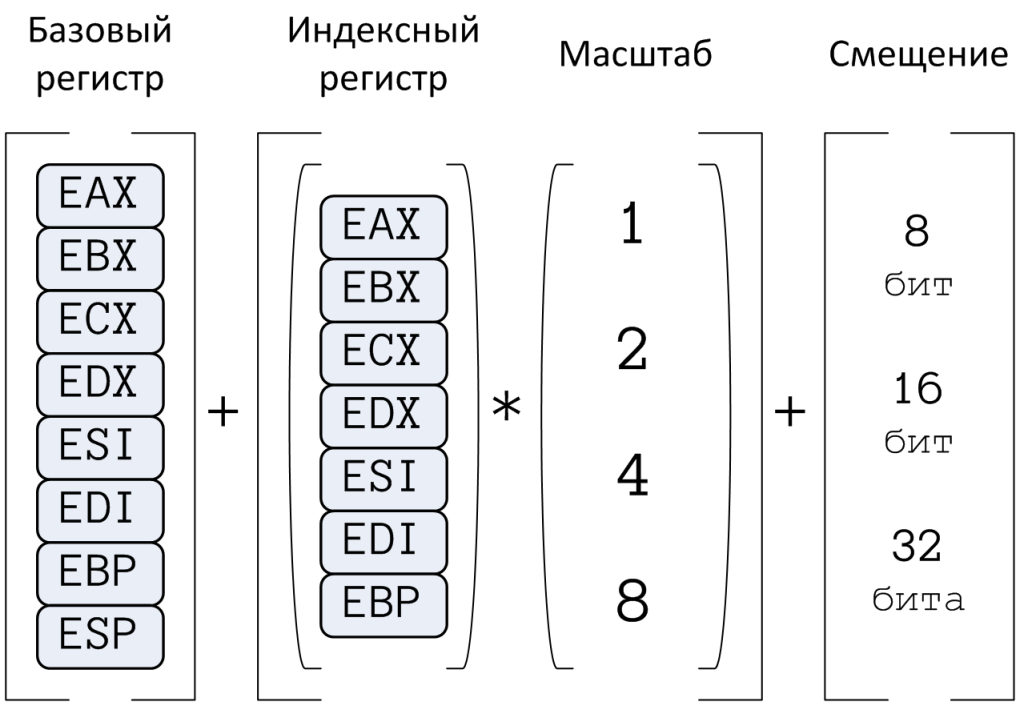
- **LEA**

Рассмотренные ранее на лекциях ассемблерные инструкции

© 2018 МГУ/ВМК/СП

Общий вид адресного кода при обращении к памяти

- LEA
  - r32, m



Исполнительный адрес = База + (Масштаб * Индекс) + Смещение

© 2018 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
```

© 2018 МГУ/ВМК/СП

Операции над отдельными битами

- Логические инструкции
  - **AND**
  - **OR**
  - **XOR**
  - **NOT**
- Сдвиги и вращения
  - **SAR**
  - **SHR**
  - **SAL, SHL**
  - **ROR**
  - **ROL**
  - **RCR**
  - **RCL**
- Битовые и байтовые
  - BT
  - BTS
  - BTR
  - BTC
  - SET*cc*
  - TEST

© 2018 МГУ/ВМК/СП

Пример

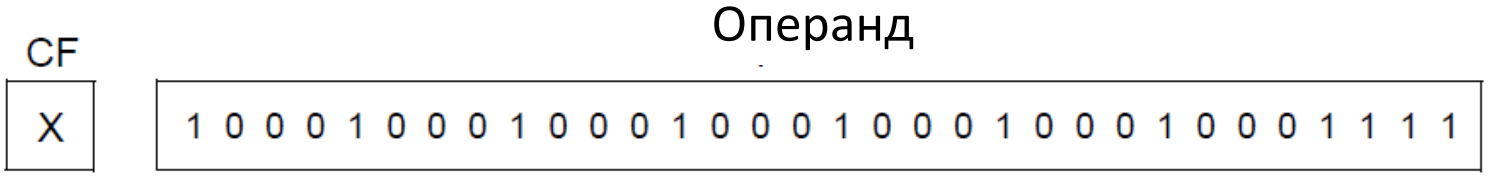
```
int pierce_arrow(int a, int b) {
   int t = ~(a | b);
   return t;
}
```

```
section .text
global pierce_arrow
pierce_arrow:
   push    ebp
   mov     ebp, esp
   mov     eax, dword [ebp+12] ; (1)
   or      eax, dword [ebp+8]  ; (2)
   not     eax                 ; (3)
   pop     ebp
   ret
```

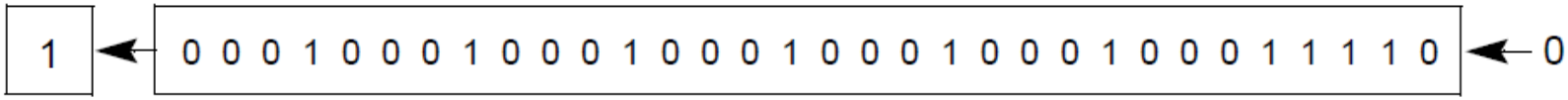
© 2018 МГУ/ВМК/СП

Сдвиг влево

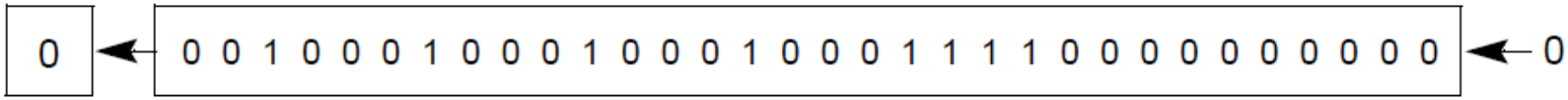
Начальное состояние



После сдвига на один разряд: SHL/SAL



После сдвига на 10 разрядов: SHL/SAL



© 2018 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

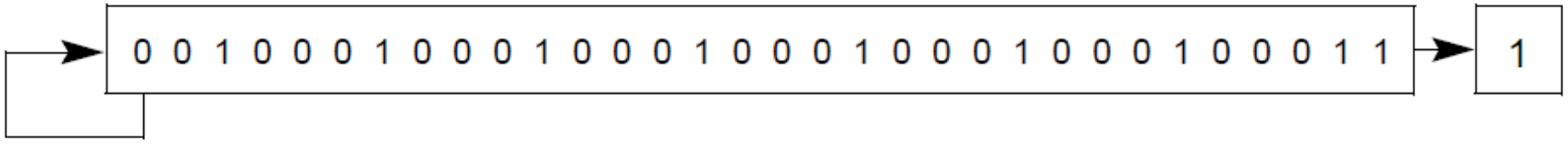
0 → 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 → 0

© 2018 МГУ/ВМК/СП

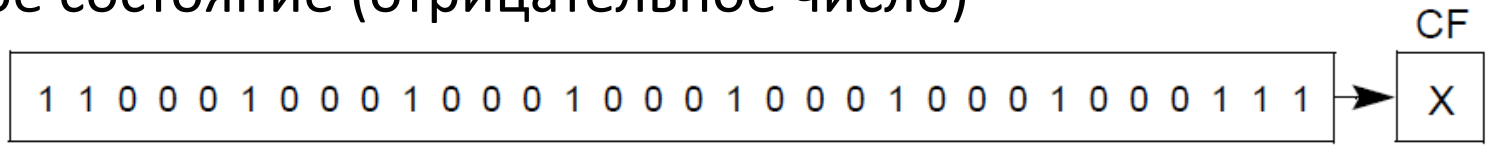
Арифметический сдвиг вправо

Начальное состояние (положительное число)

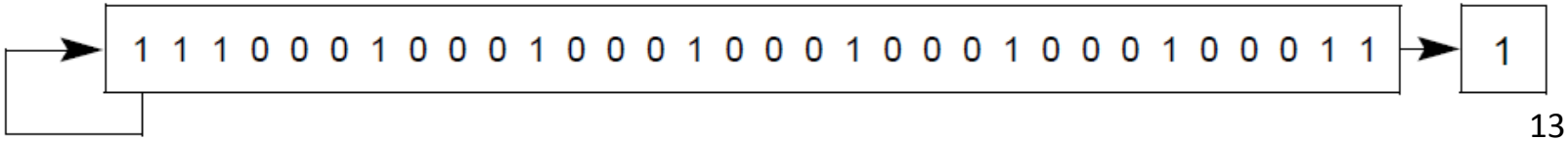
После сдвига на один разряд: SAR



Начальное состояние (отрицательное число)



После сдвига на один разряд: SAR



© 2018 МГУ/ВМК/СП

Пример

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

Что вернет `upndown(42);` ?

© 2018 МГУ/ВМК/СП

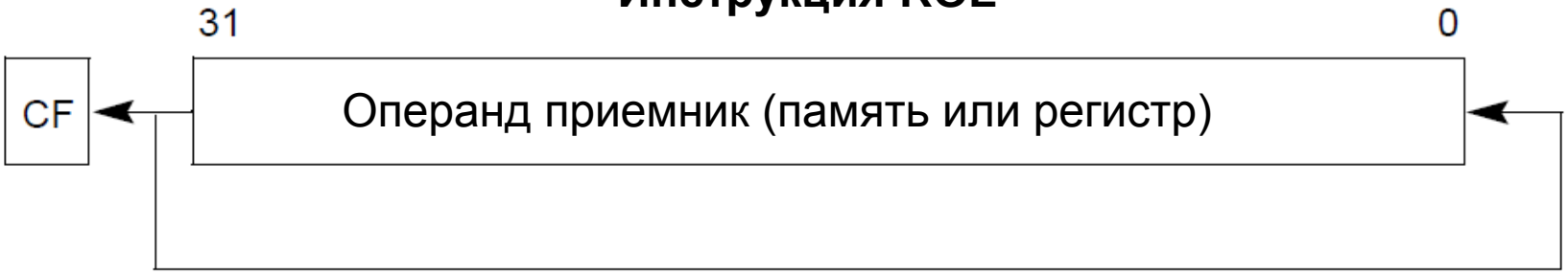
Пример

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

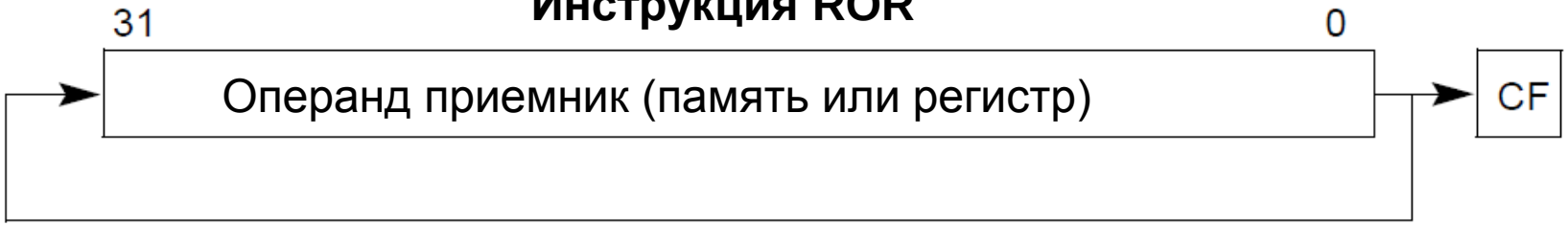
```
section .text
global upndown
upndown:
   push    ebp
   mov     ebp, esp
   movsx   eax, byte [ebp+8]
   sal     eax, 8
   sar     eax, 8
   pop     ebp
   ret
```

© 2018 МГУ/ВМК/СП

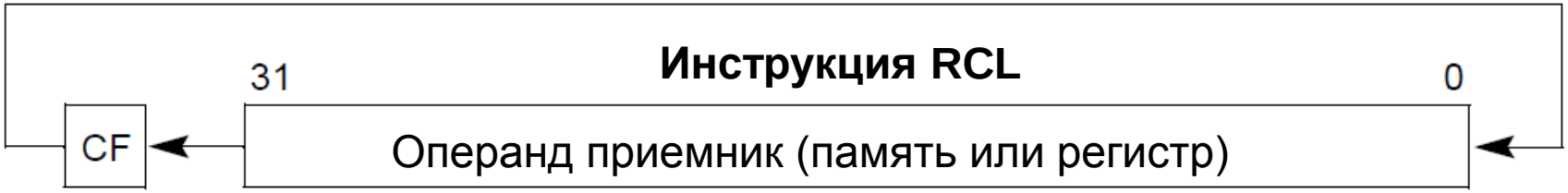
Инструкция ROL



Инструкция ROR



Инструкция RCL



Инструкция RCR

© 2018 МГУ/ВМК/СП

Пример

В 2008 году Intel были заявлены новые команды, поддерживающие шифрование AES: AESENC, AESDEC, AESENCLAST, AESKEYGENASSIST, …

```
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
  $H(M_1) = H(M_2)$