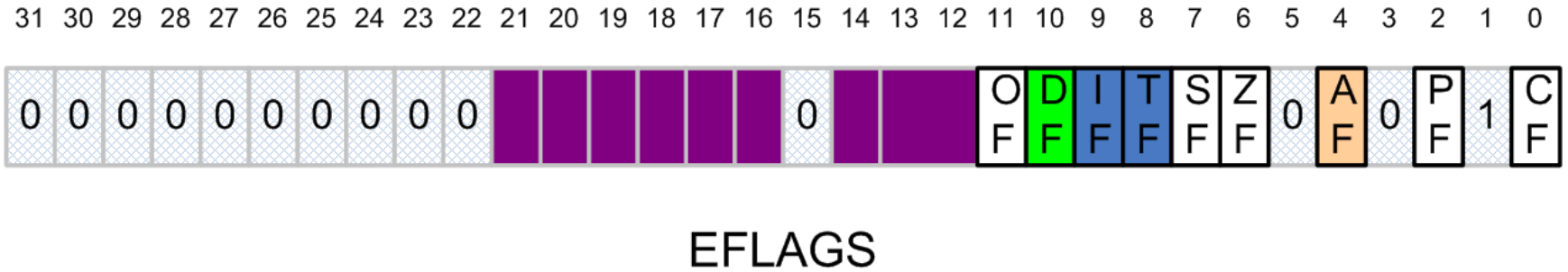
© 2016 МГУ/ВМК/СП

Лекция 4

22 февраля

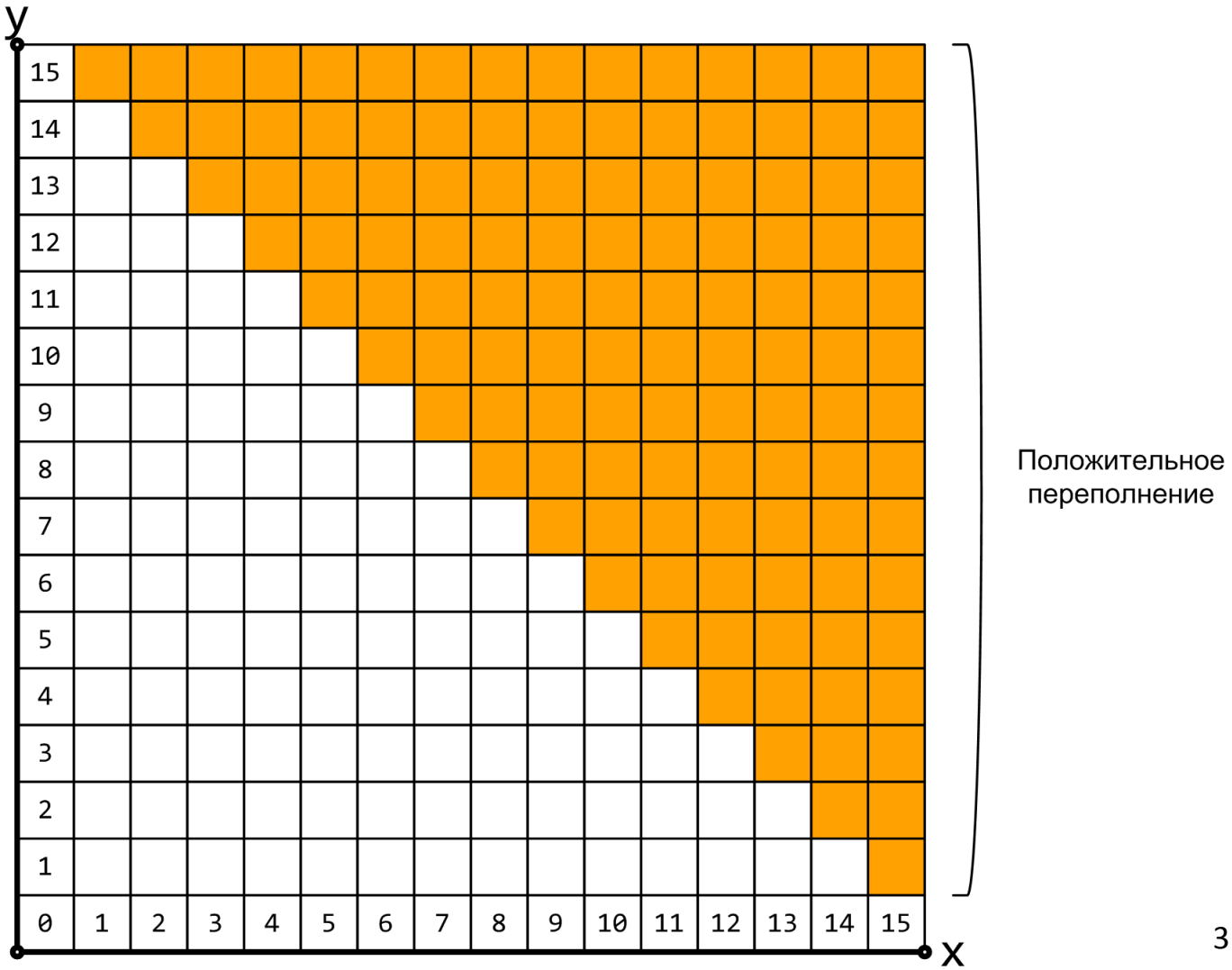
© 2016 МГУ/ВМК/СП

Регистр EFLAGS



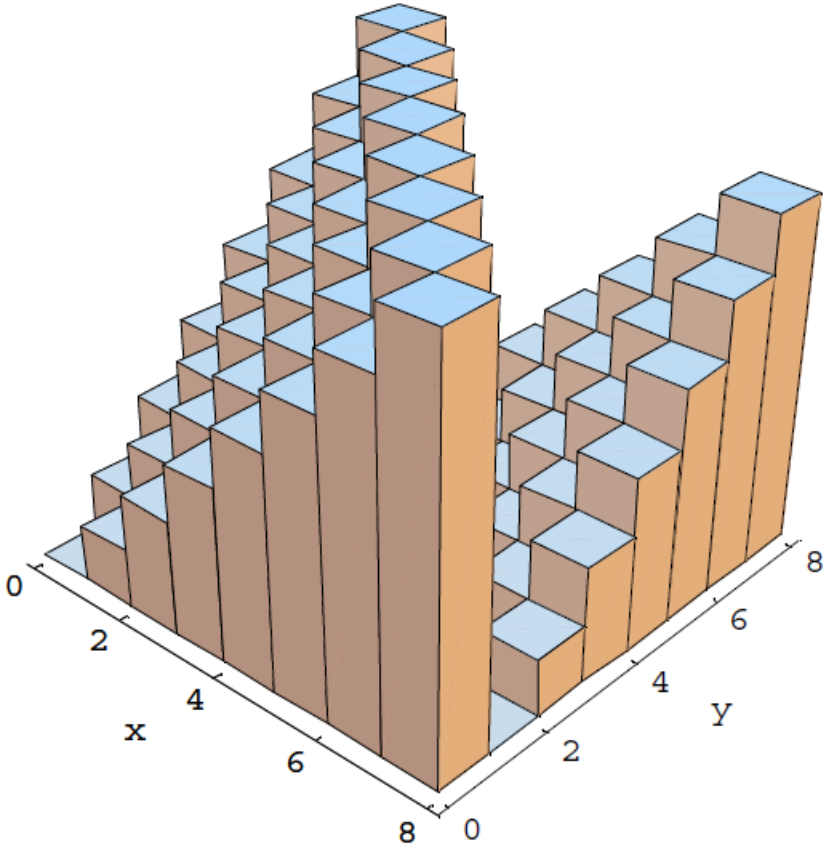
© 2016 МГУ/ВМК/СП

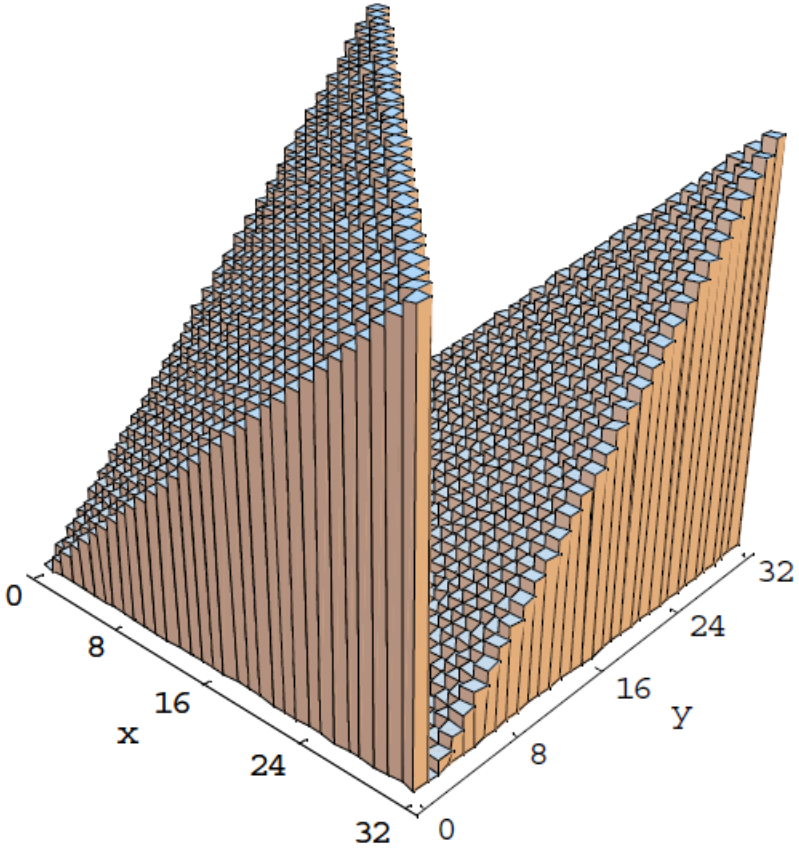
Переполнение при сложении двух беззнаковых чисел



© 2016 МГУ/ВМК/СП

Как «выглядит» переполнение в 3D?

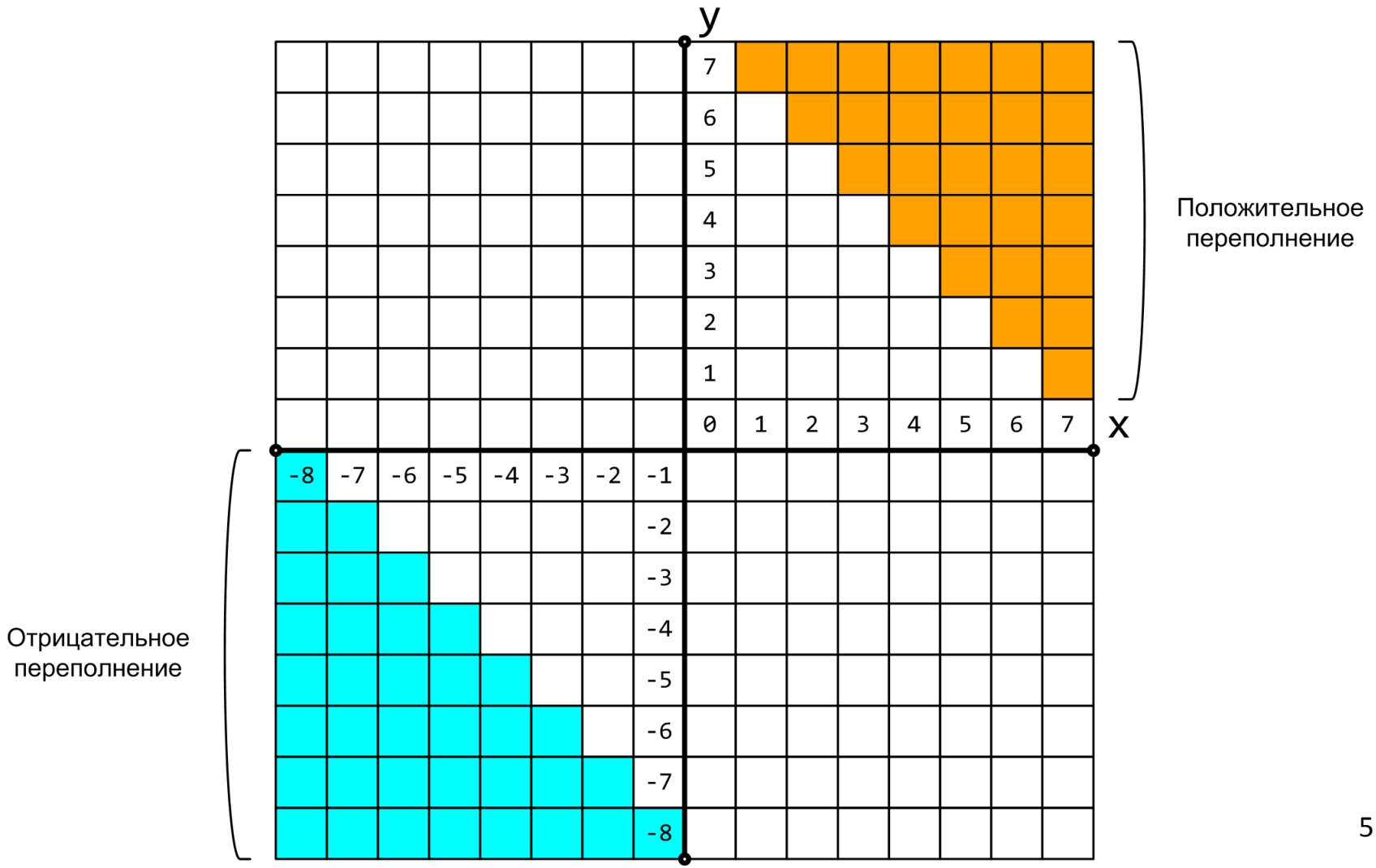




http://www.hackersdelight.org/plotsAdd.pdf

© 2016 МГУ/ВМК/СП

Переполнение при сложении двух знаковых чисел



© 2016 МГУ/ВМК/СП

Естественный порядок выполнения

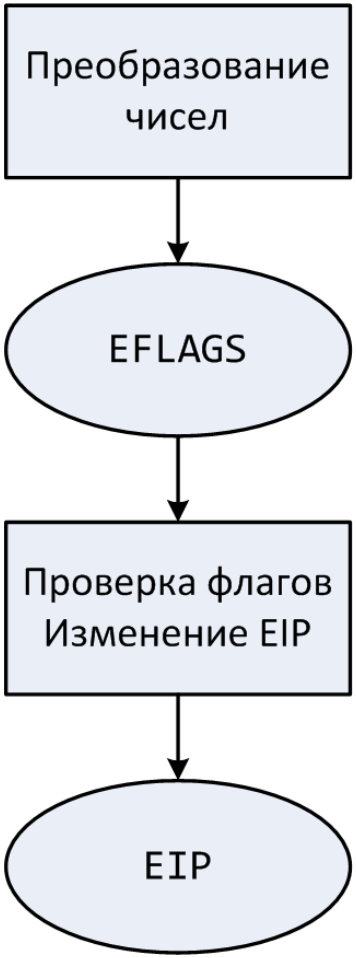
EIP	Машинный код	Длина	Ассемблерная инструкция
8048345	89 e5	2	mov ebp, esp
8048347	83 ec 10	3	sub esp, 0x10
804834a	c7 45 f0 02 00 00 00	7	mov dword [ebp-16],0x2

Прибавляем к значению регистра EIP длину в байтах декодированной команды

© 2016 МГУ/ВМК/СП

Изменение естественного порядка выполнения программы

- Арифметические операции
- CMP
  - r/m 8/16/32, i 8/16/32
  - r/m 8/16/32, r 8/16/32
  - r 8/16/32, r/m 8/16/32
- TEST
  - r/m 8/16/32, i 8/16/32
  - r/m 8/16/32, r 8/16/32
- JMP
  - r/m/i 32
- Jcc
  - i 32
- Переходы
  - Абсолютные
  - Относительные



© 2016 МГУ/ВМК/СП

Регистр EFLAGS и инструкции

	OF	SF	ZF	PF	CF
ADD, SUB, NEG	M	M	M	M	M
INC, DEC	M	M	M	M	
IMUL, MUL	M	–	–	–	M
IDIV, DIV	–	–	–	–	–
CBW, CWD, CDQ					
MOV, MOVSX, MOVZX					
CMP	M	M	M	M	M
TEST	0	M	M	M	0

«M» инструкция обновляет флаг (сбрасывает или устанавливает)
«–» влияние инструкции на флаг не определено
« » инструкция на флаг не влияет
«0» инструкция сбрасывает флаг

© 2016 МГУ/ВМК/СП

```
...
SECTION .text
GLOBAL CMAIN
CMAIN:
   MOV  EAX, DWORD [a]     ; (1)
   TEST EAX, EAX           ; (2)
   JE   .1                 ; (3)
   MOV  ECX, DWORD [b]     ; (4)
   TEST ECX, ECX           ; (5)
   JE   .1                 ; (6)
   CDQ                     ; (7)
   IDIV ECX                ; (8)
   SUB  DWORD [a], EDX     ; (9)
 .1:
   XOR  EAX, EAX           ; (10)
   RET                     ; (11)
```

© 2016 МГУ/ВМК/СП

Просмотр содержимого исполняемого файла

```
-bash-2.05b$ ./build_asm.sh backward.asm
-bash-2.05b$ objdump -d -M intel backward
```

```
080483e0 <main>:
 80483e0:       a1 fc 94 04 08          mov    eax,ds:0x80494fc
 80483e5:       85 c0                   test   eax,eax
 80483e7:       74 13                   je     80483fc <main.1>
 80483e9:       8b 0d 00 95 04 08       mov    ecx,ds:0x8049500
 80483ef:       85 c9                   test   ecx,ecx
 80483f1:       74 09                   je     80483fc <main.1>
 80483f3:       99                      cdq
 80483f4:       f7 f9                   idiv   ecx
 80483f6:       29 15 fc 94 04 08       sub    ds:0x80494fc,edx

080483fc <main.1>:
 80483fc:       31 c0                   xor    eax,eax
 80483fe:       c3                      ret
 80483ff:       90                      nop
```

© 2016 МГУ/ВМК/СП

Далее…

- Архитектура фон Неймана
- Архитектура IA-32
- Вычисление арифметических выражений
- Передача управления
- **Организация вызова функций**

© 2016 МГУ/ВМК/СП

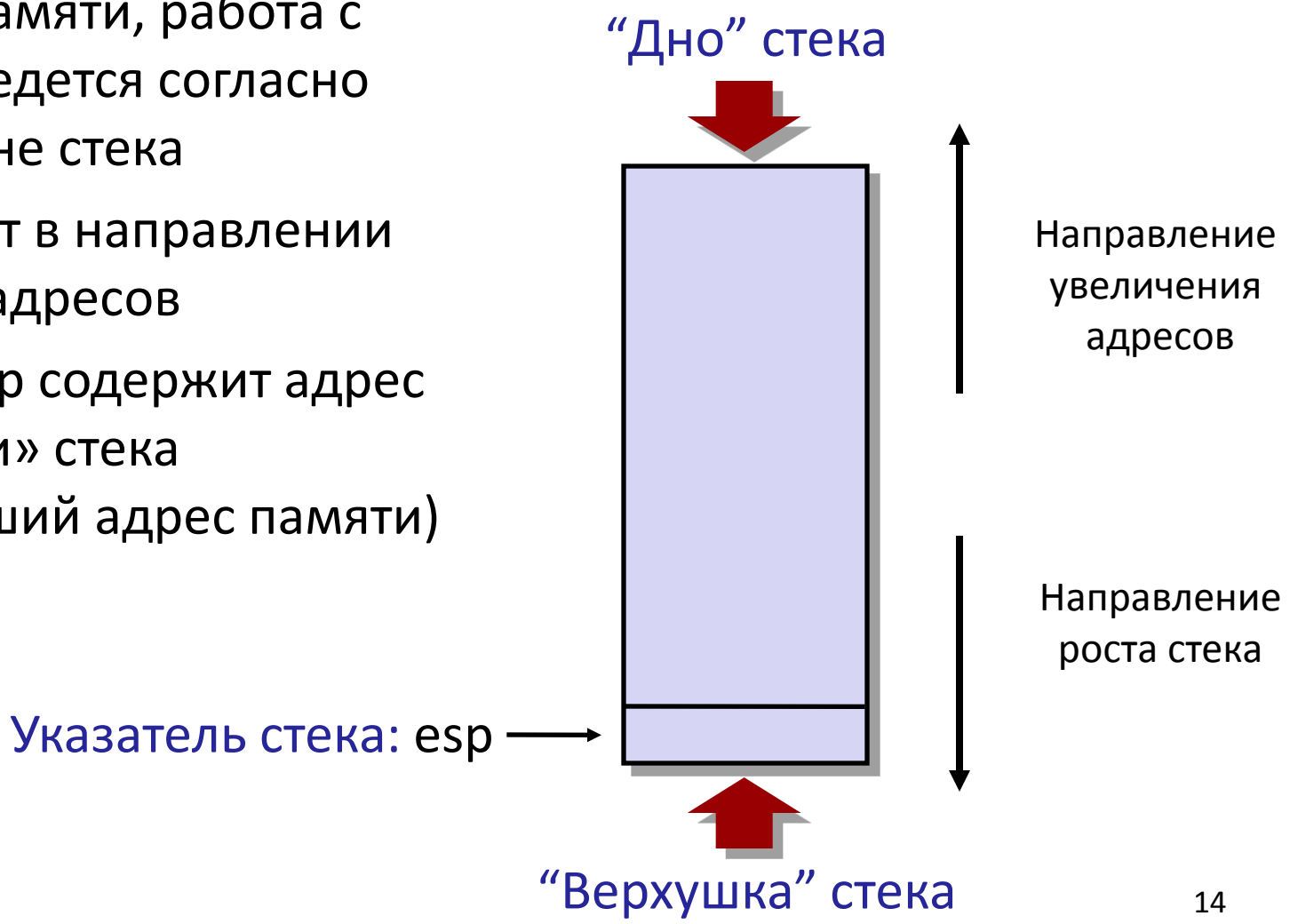
Организация вызова функций

- Вопросы
  - Передача управления и возвращение обратно
  - Вычисление значений фактических параметров и их размещение
  - Передача возвращаемого значения
  - Размещение автоматических локальных переменных
  - Порядок использование регистрового файла различными функциями
  - Какие именно машинные команды использовать для поддержки функций
- Ответы – Application Binary Interface (ABI)
  - Соглашение о вызовах (Calling Convention)

© 2016 МГУ/ВМК/СП

Аппаратный стек IA-32

- Область памяти, работа с которой ведется согласно дисциплине стека
- Стек растет в направлении меньших адресов
- Регистр esp содержит адрес «верхушки» стека (наименьший адрес памяти)

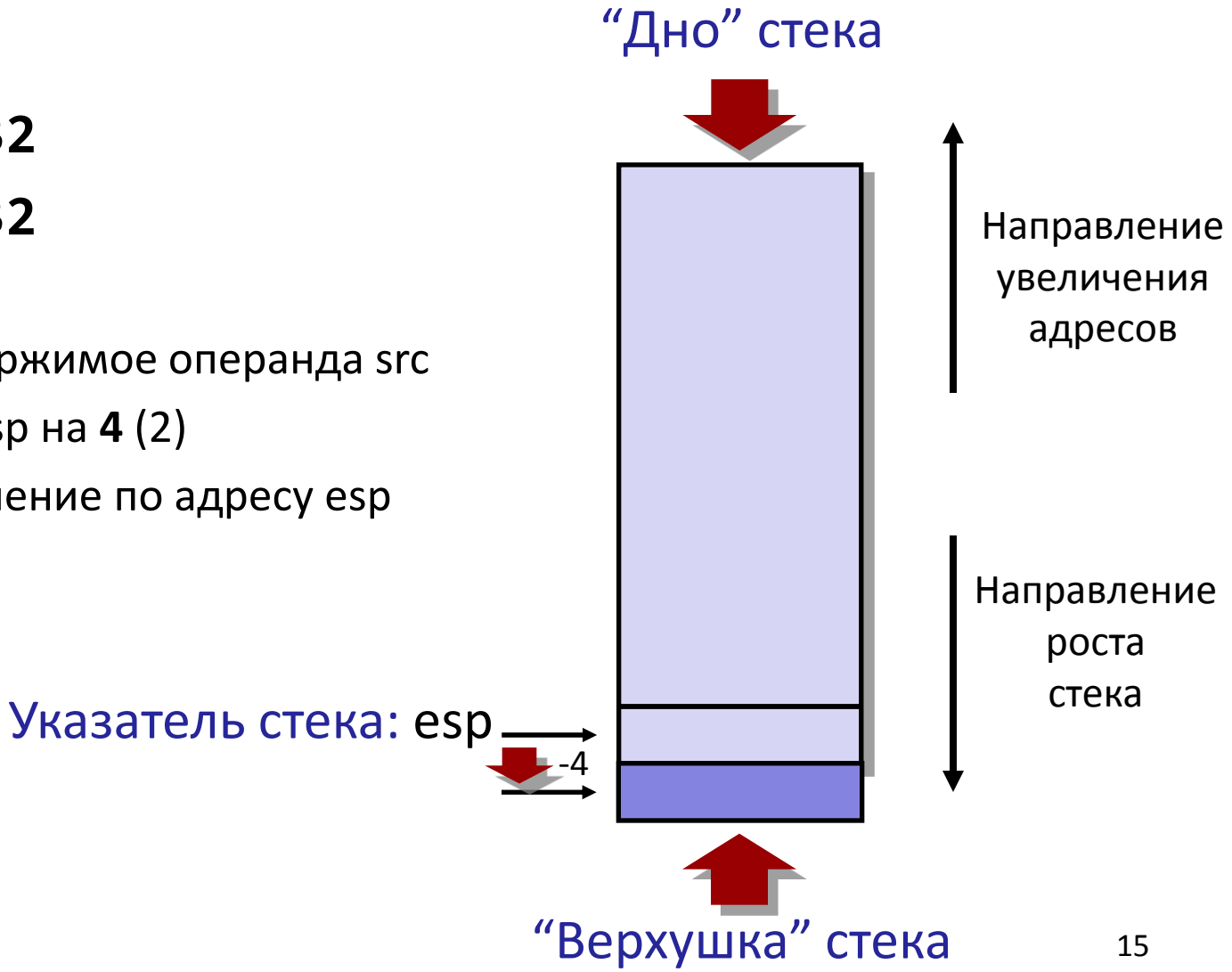


© 2016 МГУ/ВМК/СП

Загрузка данных в стек: Push

- `push src`
  - `r/m 16/32`
  - `i 8/16/32`

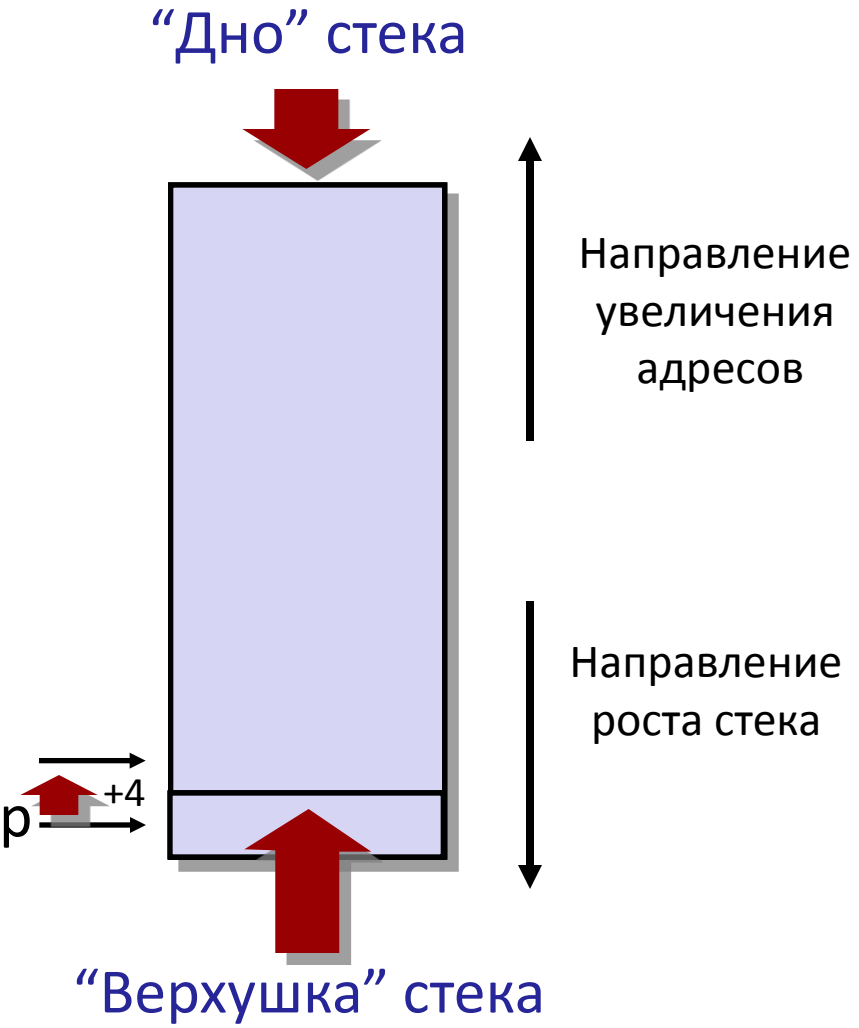
  – Извлечь содержимое операнда src
  – Уменьшить esp на **4** (2)
  – Записать значение по адресу esp



© 2016 МГУ/ВМК/СП

Выгрузка данных из стека: Pop

- `pop dst`
  - `r/m 16/`**`32`**
  - Извлечь значение по адресу esp
  - Увеличить esp на **4** (2)
  - Записать содержимое операнда dst



© 2016 МГУ/ВМК/СП

Языки программирования (ЯП), базирующиеся на стеке вызовов

- ЯП с поддержкой рекурсии
  - C, Pascal, Java
  - Код функции можно вызывать повторно (“Reentrant”)
    - Одновременно могут выполняться несколько вызовов функции
  - Необходимо выделять память под сохранение состояния каждого работающего вызова
    - Аргументы
    - Локальные переменные
    - Адрес возврата
- Стек
  - Сохранять состояние вызова функции надо в ограниченный период времени: от момента вызова до момент выхода
  - Вызываемая функция всегда завершается до вызывающей
- Стек выделяется **Фреймами**
  - Состояние отдельного вызова функции