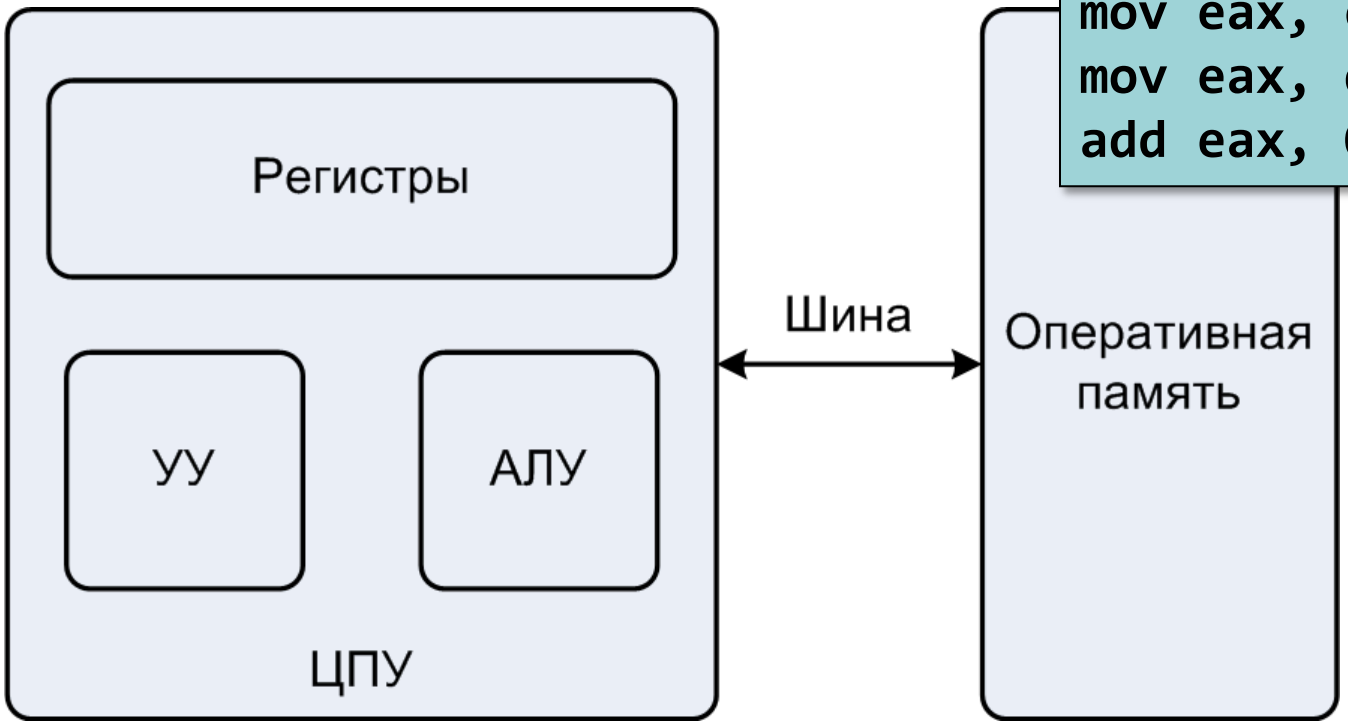
© 2014 МГУ/ВМК/СП

Лекция 2

12 февраля

© 2014 МГУ/ВМК/СП

Архитектура фон Неймана



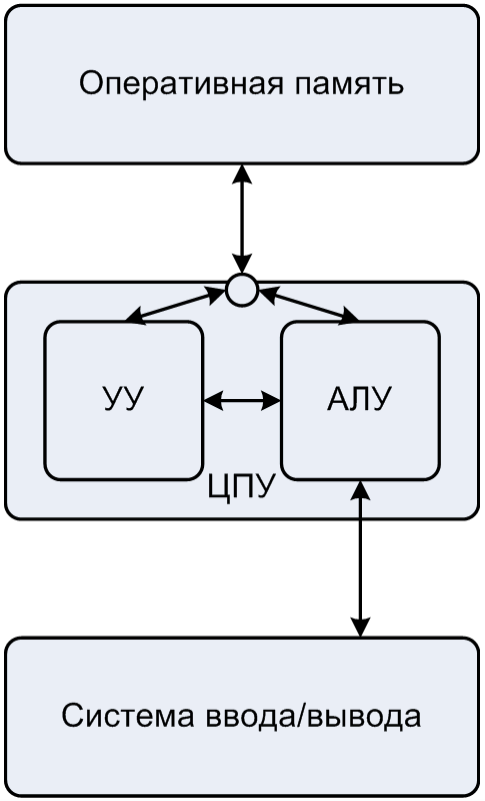

```
mov eax, ecx
mov eax, dword [0x2014]
add eax, 0xff
```

1. Двоичное кодирование информации
2. Неразличимость команд и данных
3. Адресуемость памяти
4. Последовательное выполнение команд

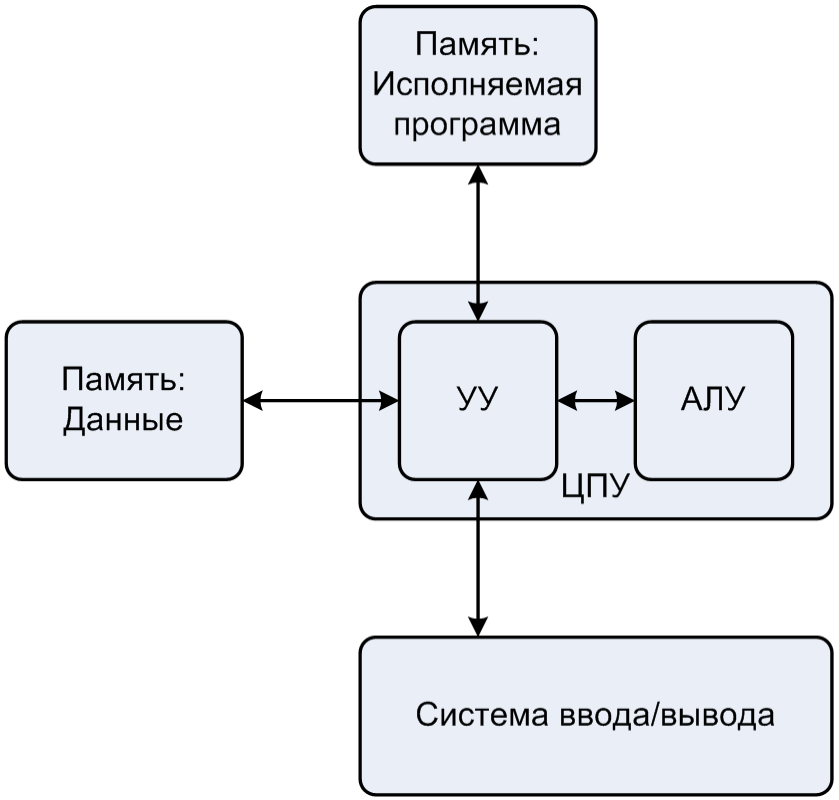
Вспоминаем Си-машину…

© 2014 МГУ/ВМК/СП

Единственный вариант архитектуры?



Альтернативный способ изображения архитектуры фон Неймана



Гарвардская архитектура

Вспоминаем Си-машину…

© 2014 МГУ/ВМК/СП

Цикл работы

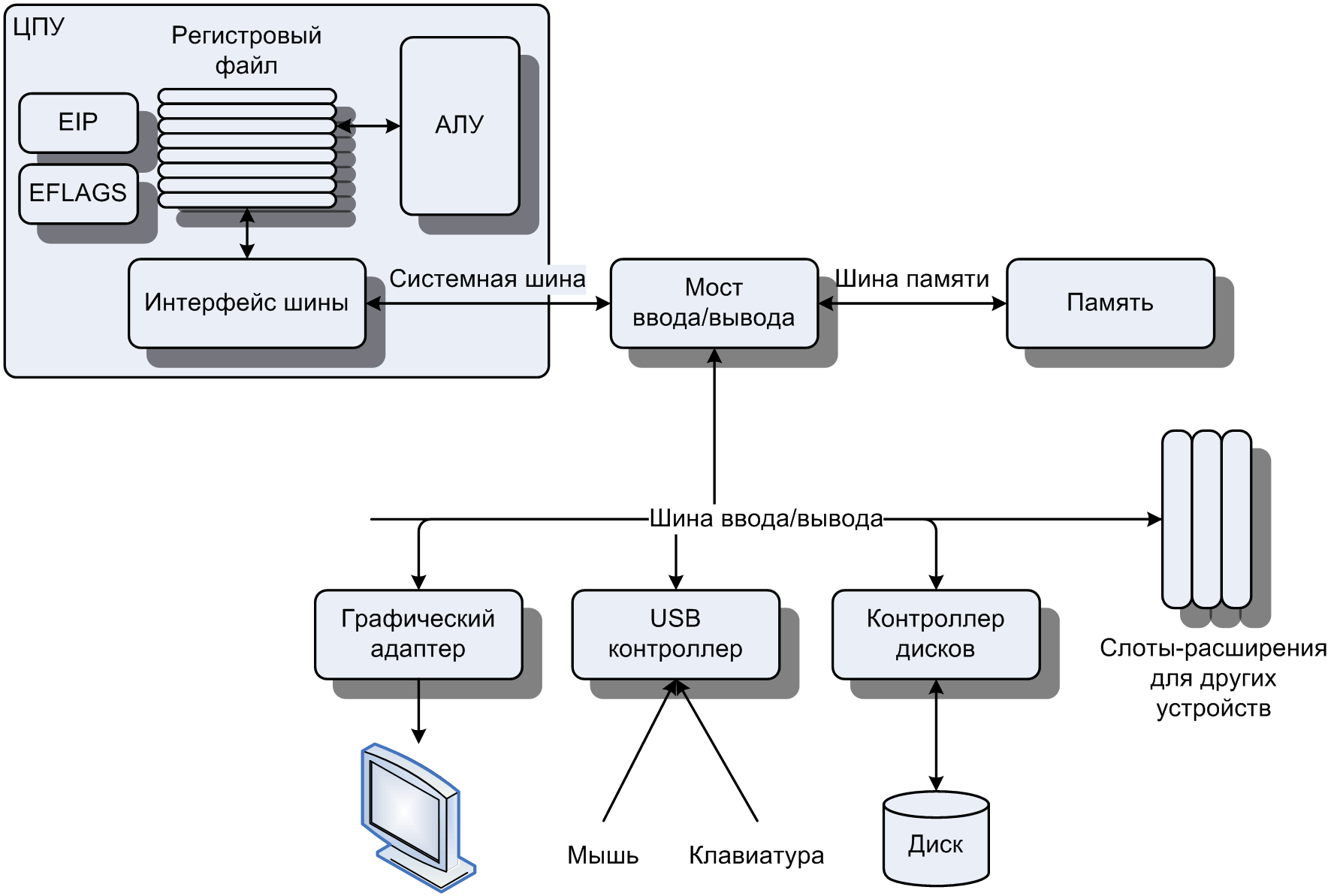
1. **Извлечение инструкции из памяти**
   Используя текущее значение *счетчика команд,* процессор извлекает некоторое количество байт из памяти и помещает их в *буфер команд*
2. **Декодирование команды**
   Процессор просматривает содержимое *буфера команд* и определяет *код операции* и ее *операнды*
   Длина декодированной команды прибавляется к текущему значению *счетчика команд*
3. **Загрузка операндов**
   Извлекаются значения операндов
   Если операнд размещен в ячейках памяти – вычисляется *исполнительный адрес*
4. **Выполнение операции над данными**
5. **Запись результата**
   Результат может быть записан в том числе и в *счетчик команд* для изменения естественного порядка выполнения

© 2014 МГУ/ВМК/СП

Промежуточные итоги: ключевые термины и понятия

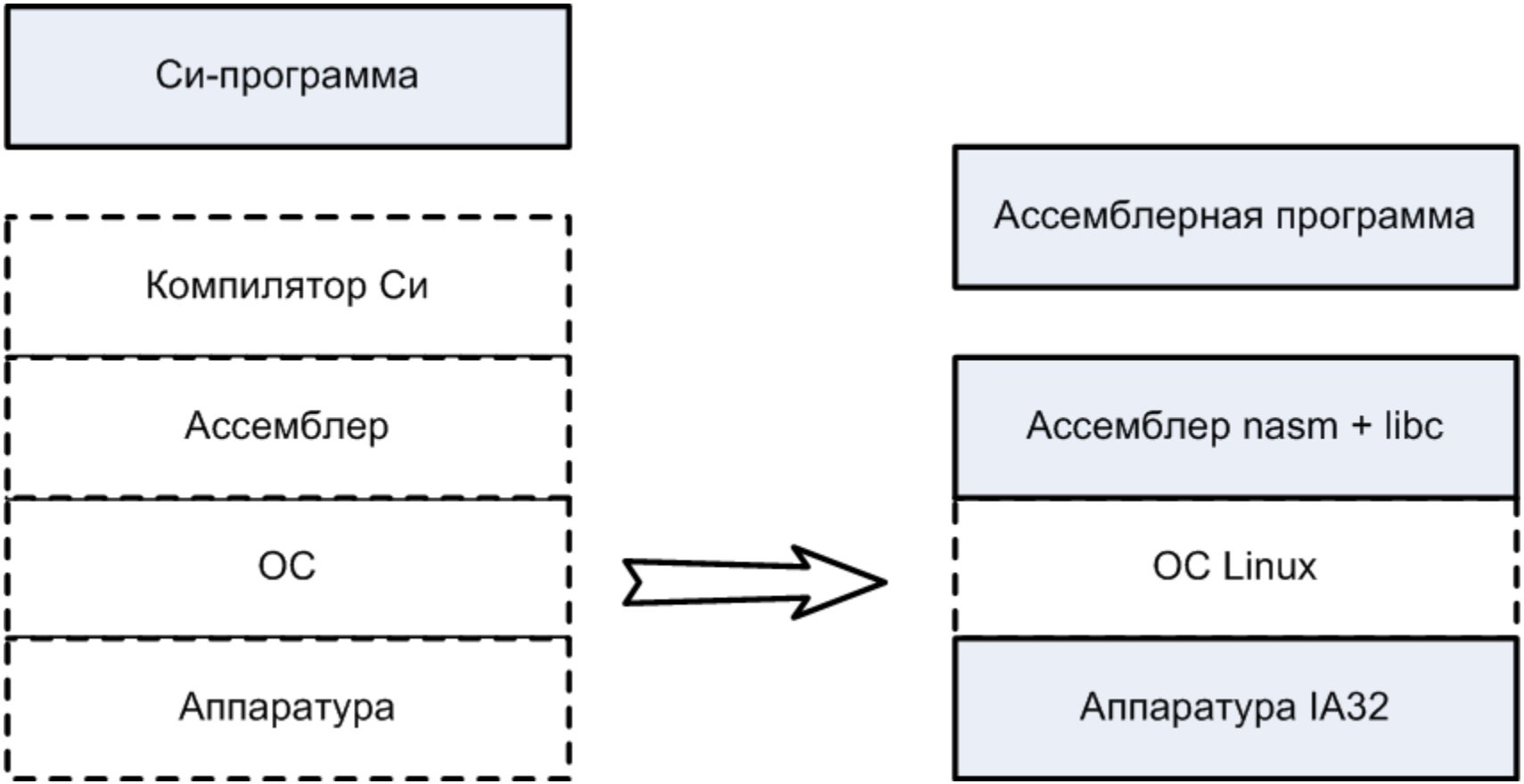
- Регистр, имя регистра, значение регистра, разрядность регистра, машинное слово
- Память, ячейка памяти, адрес ячейки памяти, время доступа к памяти
- Цикл, тактовая частота, счетчик команд
- Машинная команда, код операции, операнд, адресность команды
  Исполнительный (действительный) адрес
  англ. *effective (executive) address*
- Способ адресации
  прямая, непосредственная, косвенная

© 2014 МГУ/ВМК/СП



© 2014 МГУ/ВМК/СП

Окружение разработки и выполнения пользовательской программы

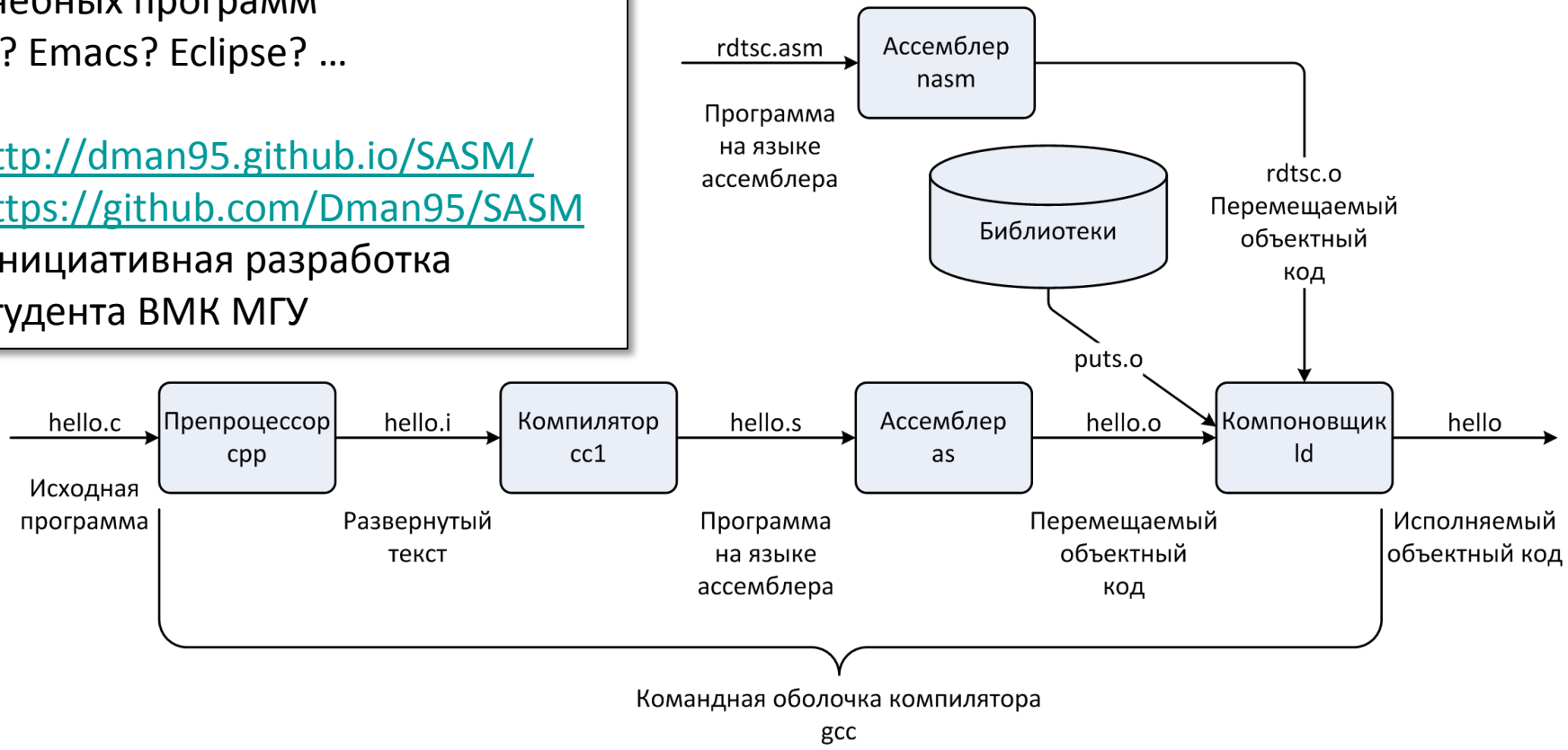


© 2014 МГУ/ВМК/СП

Этапы компиляции и компоновки пользовательской программы

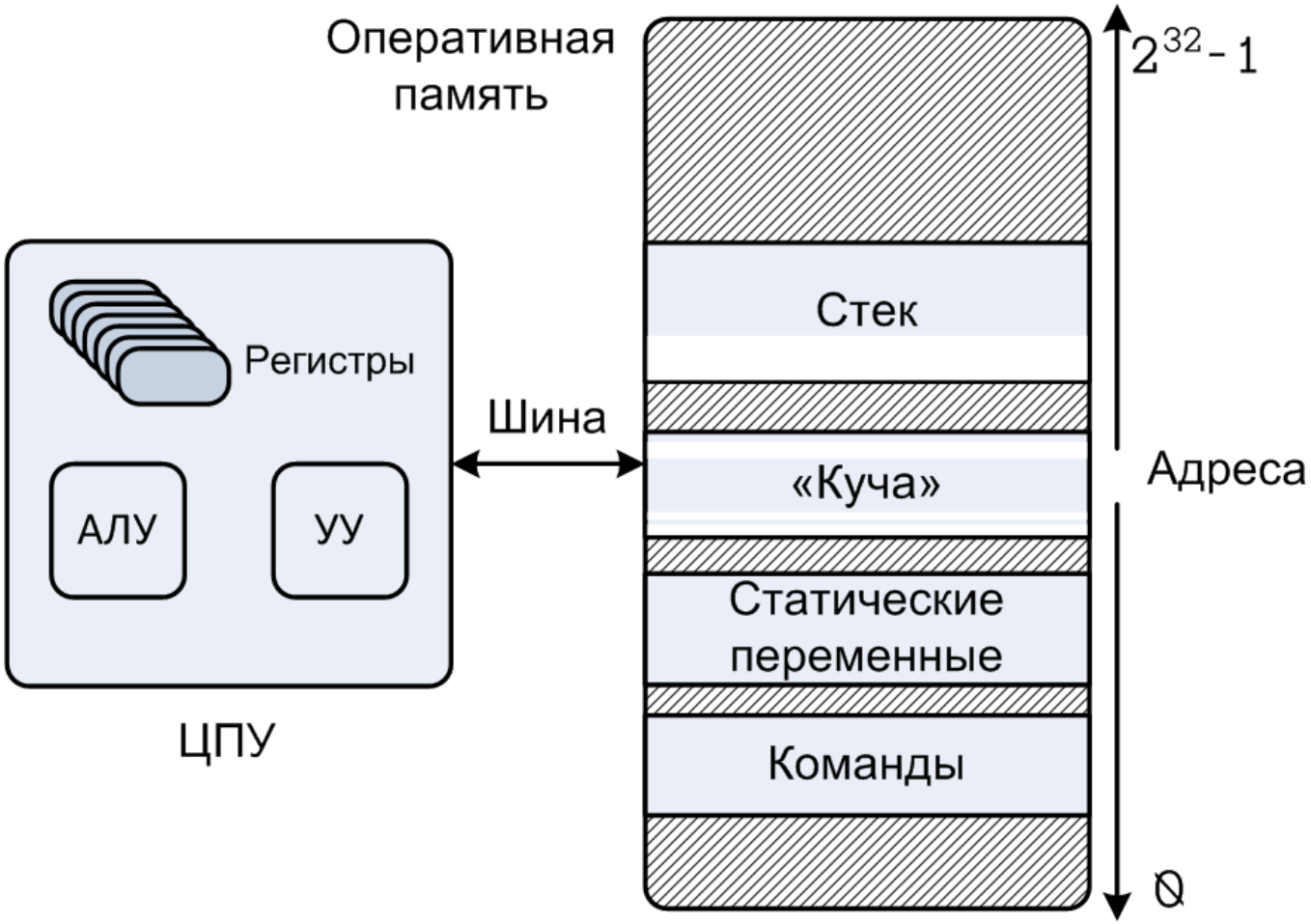
Удобная среда разработки для учебных программ
vi? Emacs? Eclipse? …

http://dman95.github.io/SASM/
https://github.com/Dman95/SASM
Инициативная разработка студента ВМК МГУ



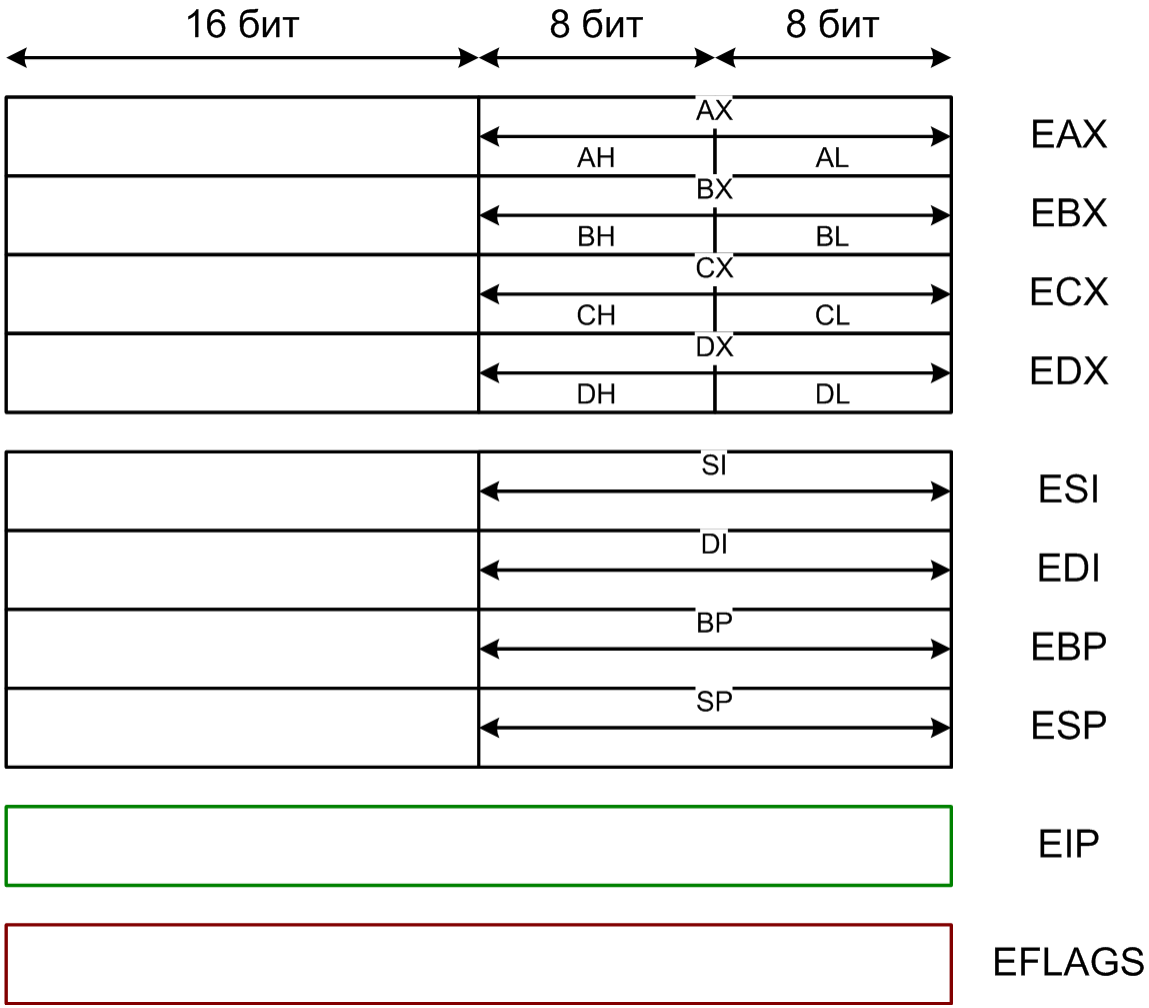
© 2014 МГУ/ВМК/СП

Машина, на которой работает пользовательская программа (архитектура IA-32)



Вспоминаем Си-машину…

© 2014 МГУ/ВМК/СП



Типы операндов

- Регистр r
- Память m
- Константа c

Размеры операндов

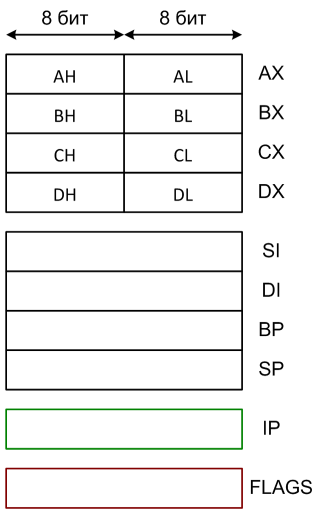
- byte 8
- word 16
- dword 32
- qword 64

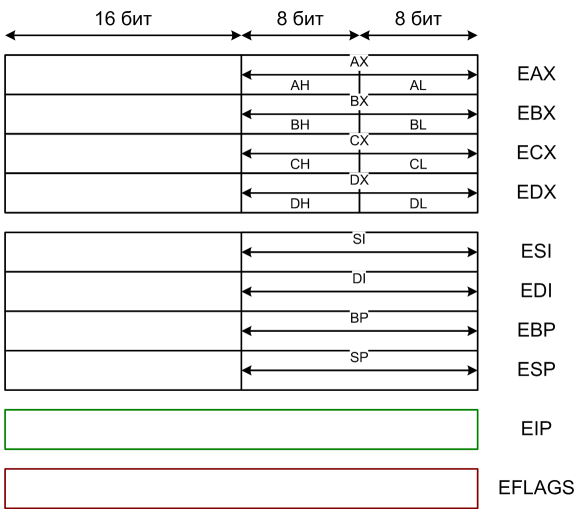
```
mov eax, ecx
mov eax, dword [0x2014]
add eax, 0xff
mov dword [ebp-4], eax
```

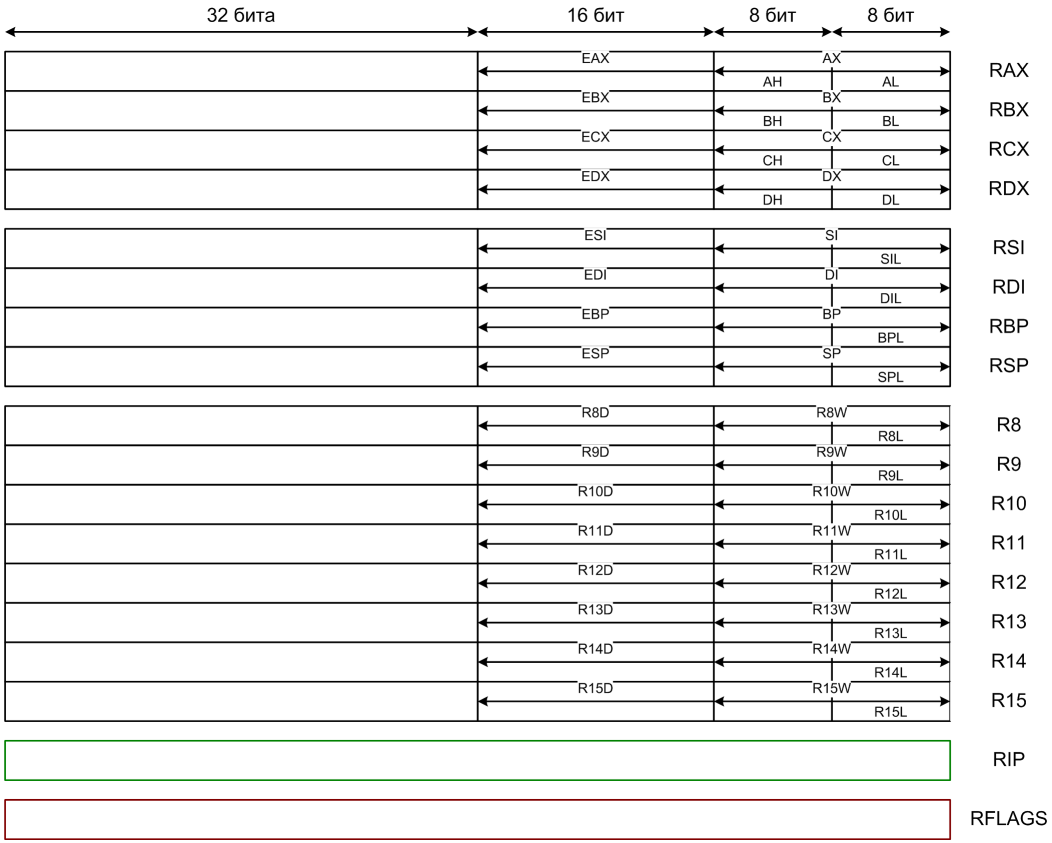
Где сегментные регистры?

© 2014 МГУ/ВМК/СП

8086 → IA-32→ Intel64





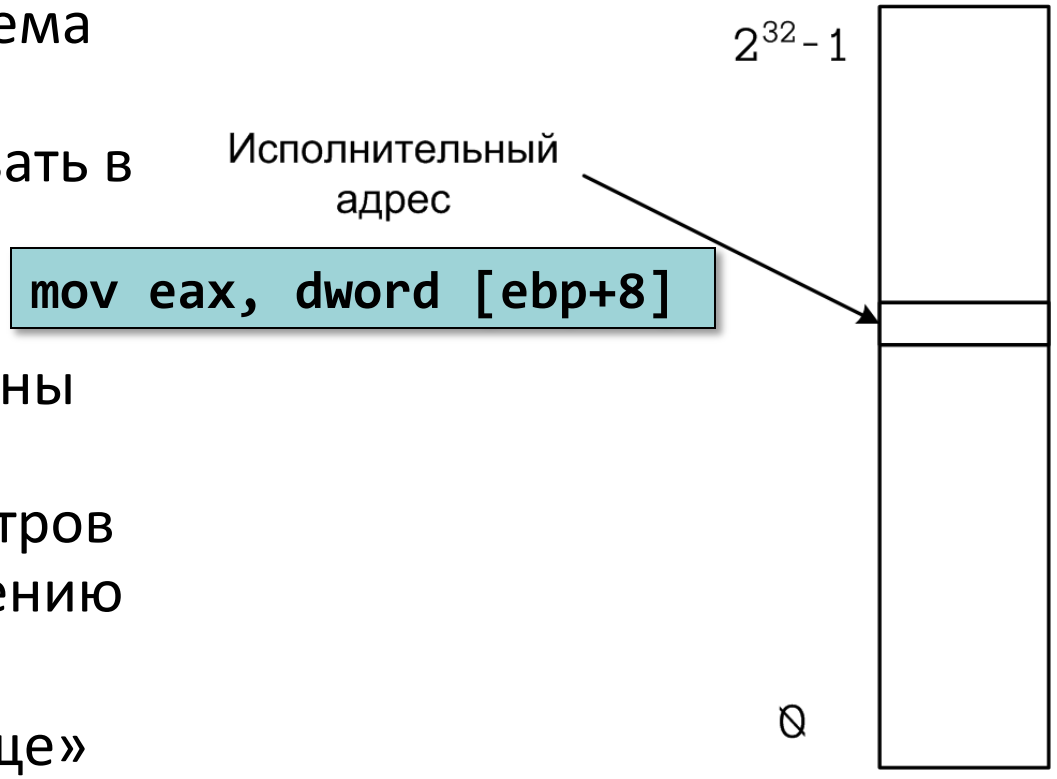


Какие еще названия встречаются?
x86, x86_64, AMD64
IA-64 совсем другая архитектура

© 2014 МГУ/ВМК/СП

Особенности IA-32?

- Особенности аппаратуры и операционная система (Windows/Linux) позволяют использовать в программах модель плоской памяти
- Значительно ослаблены ограничения на использование регистров в командах по сравнению с 8086
- «Количественно проще» чем Intel64



© 2014 МГУ/ВМК/СП

```c
void f() {
   static int cntr = 0;      // 1
   int x = 2, y = 1, z = 0; // 2
   unsigned short w = 282;  // 3
   signed char q = 13;      // 4
   ++cntr;                  // 5
   z = -x + q * w *y - w;   // 6
}
```

```
section .bss
; Резервирование 4 байт памяти
   cntr  resd  1
section .text
   global f
; Точка входа в программу
f:
     push    ebp
     mov     ebp, esp
     sub     esp, 16
     mov     dword [ebp-16], 2     ; (1)
     mov     dword [ebp-12], 1     ; (2)
     mov     dword [ebp-8], 0      ; (3)
     mov     word  [ebp-4], 282    ; (4)
     mov     byte  [ebp-1], 13     ; (5)
     add     dword [cntr], 1       ; (6)
     movsx   eax, byte [ebp-1]     ; (7)
     movzx   edx, word [ebp-4]     ; (8)
     imul    eax, edx              ; (9)
     imul    eax, dword [ebp-12]   ; (10)
     sub     eax, dword [ebp-16]   ; (11)
     sub     eax, edx              ; (12)
     mov     dword [ebp-8], eax    ; (13)
     leave
     ret
```

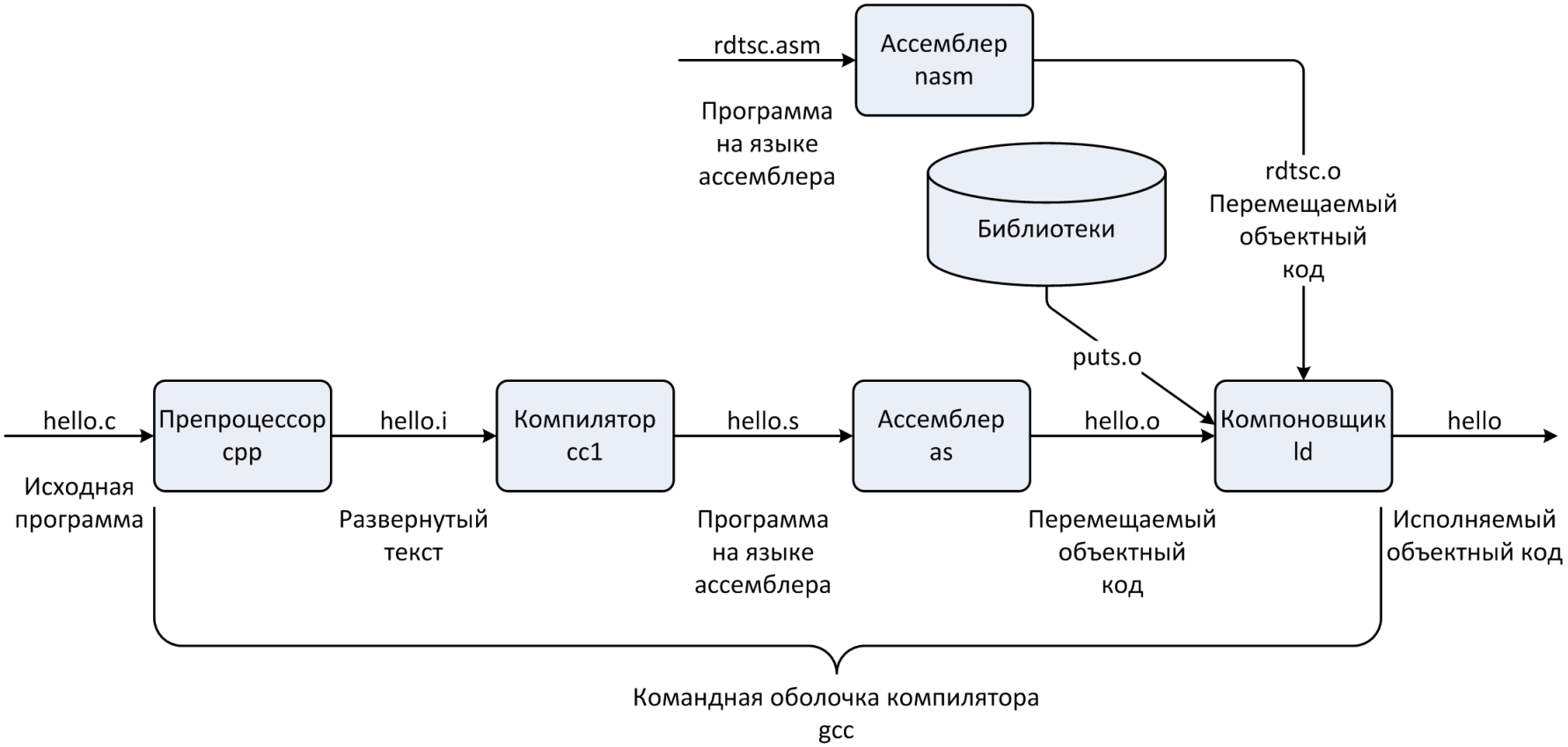
© 2014 МГУ/ВМК/СП

```
snoop@earth:~/samples$ nasm -f elf -o sample1.o sample1.asm
snoop@earth:~/samples$ objdump -M intel -d sample1.o
```

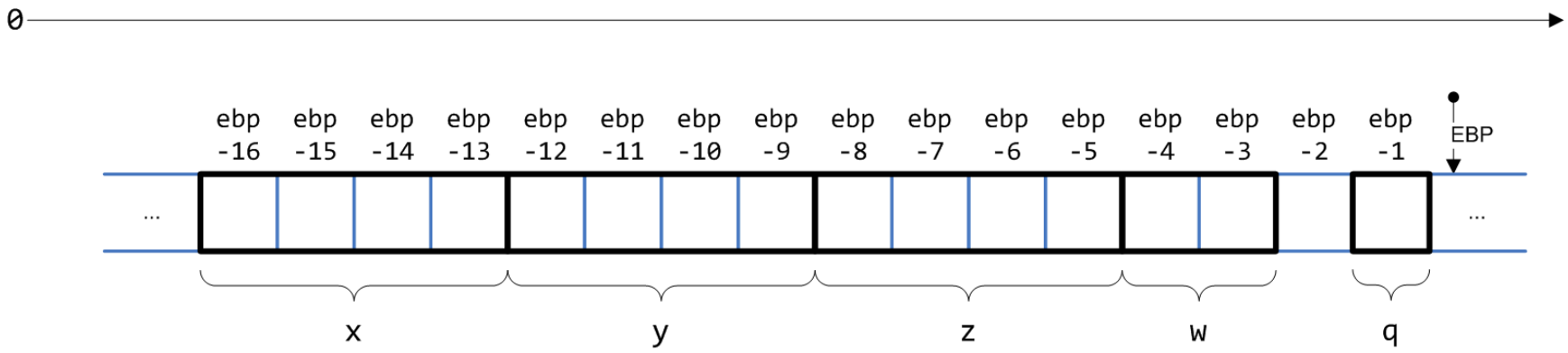
```
   0:   55                      push   ebp
   1:   89 e5                   mov    ebp,esp
   3:   81 ec 10 00 00 00       sub    esp,0x10
   9:   c7 45 f0 02 00 00 00    mov    DWORD PTR [ebp-0x10],0x2
  10:   c7 45 f4 01 00 00 00    mov    DWORD PTR [ebp-0xc],0x1
  17:   c7 45 f8 00 00 00 00    mov    DWORD PTR [ebp-0x8],0x0
  1e:   66 c7 45 fc 1a 01       mov    WORD PTR [ebp-0x4],0x11a
  24:   c6 45 ff 0d             mov    BYTE PTR [ebp-0x1],0xd
  28:   81 05 00 00 00 00 01    add    DWORD PTR ds:0x0,0x1
  2f:   00 00 00
  32:   0f be 45 ff             movsx  eax,BYTE PTR [ebp-0x1]
  36:   0f b7 55 fc             movzx  edx,WORD PTR [ebp-0x4]
  3a:   0f af c2                imul   eax,edx
  3d:   0f af 45 f4             imul   eax,DWORD PTR [ebp-0xc]
  41:   2b 45 f0                sub    eax,DWORD PTR [ebp-0x10]
  44:   29 d0                   sub    eax,edx
  46:   89 45 f8                mov    DWORD PTR [ebp-0x8],eax
  49:   c9                      leave
  4a:   c3                      ret
```

Синтаксис objdump отличается от nasm!

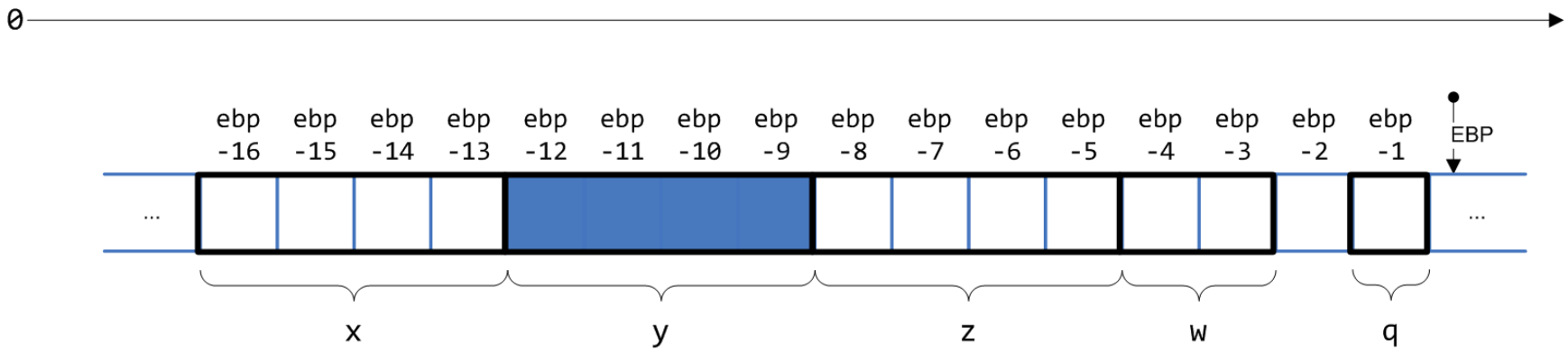
© 2014 МГУ/ВМК/СП



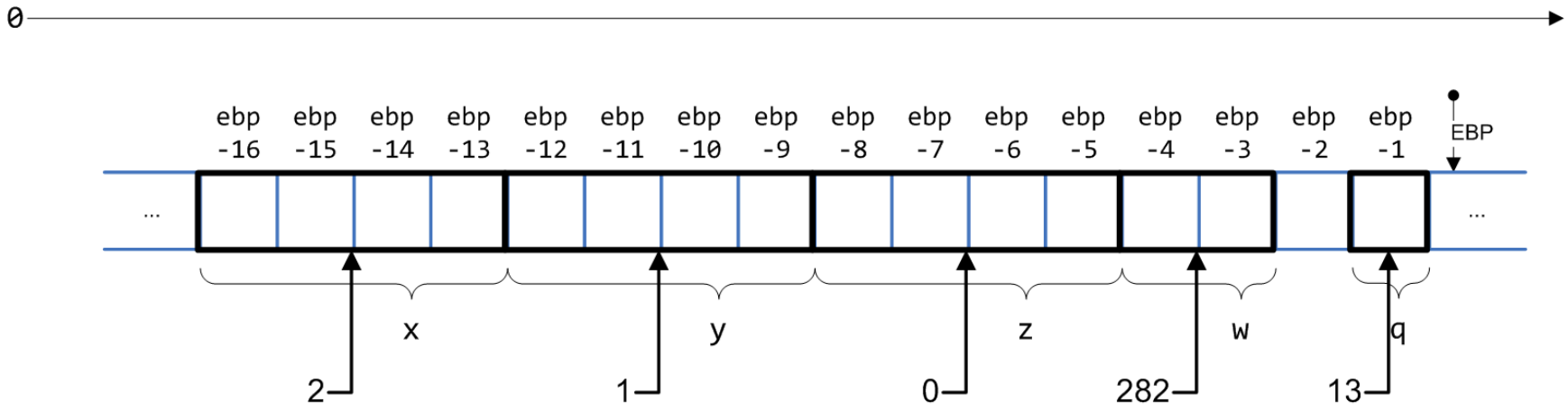
© 2014 МГУ/ВМК/СП



© 2014 МГУ/ВМК/СП



© 2014 МГУ/ВМК/СП



© 2014 МГУ/ВМК/СП

```
%include 'io.inc'

section .data
 a    dw 1
 addr dd $
 var  dd 0x1234F00D

section .bss
   cntr resd 1

section .text
global CMAIN
CMAIN:
   add dword [cntr], 1
   mov eax, [addr]
   PRINT_HEX 4, eax
   NEWLINE
   PRINT_HEX 4, addr
   NEWLINE
   xor eax, eax
```