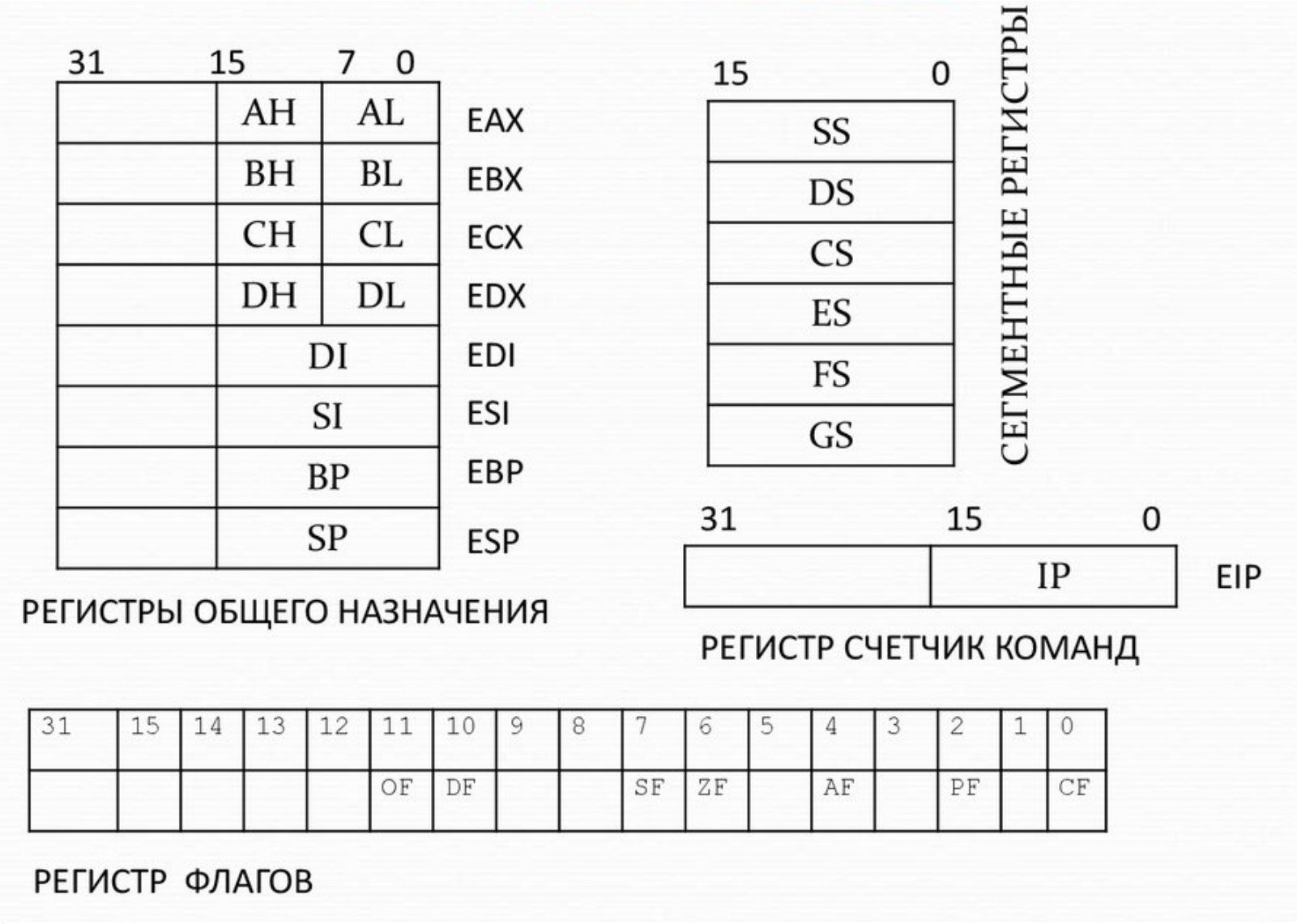
Языки программирования (Asm)

Занятие №2. Циклы. Безусловный и условные переходы.

МИРЭА - РТУ, БК №252

Москва 2020

Регистры процессоров архитектуры x86



Стек


Оперативная память
0000 : 0000
Сегмент кода
Сегмент данных
Сегмент стека
ss : sp
Вершина стека
Направление роста стека
ss : ffff
Дно стека
Старшие адреса оперативной памяти

Использование стека

Опуская подробности, стек представляет собой область памяти, в которую можно записывать временные данные без использования переменных. В стеке данные хранятся строго последовательно, и могут быть взяты оттуда только в обратной последовательности.
Допустим, мы хотим восстановить значения регистров ECX и EDX после вывода на экран строки. Стек позволяет это сделать.

```
1  push ecx
2  push edx
3
4  mov eax, 4
5  mov ebx, 1
6  mov ecx, hello
7  mov edx, len
8  int 0x80
9
10 pop edx
11 pop ecx
```

Использование стека

Команда PUSH "заносит"значение нужного регистра в стек, а команда POP "достает"это значение. Если мы положили в стек значения регистров EAX, EBX, ECX, то доставать их нужно в обратной последовательности ECX, EBX, EAX.
В случае написания программы для 64-разрядной архитектуры, в стек нужно будет заносить 64-разрядные регистры (RAX, RBX, RCX и т.д.) вместо 32-разрядных.
Стек удобно использовать для восстановления значений регистров после их использования для других целей. Например, 4 основных регистра общего назначения используются при операциях ввода/вывода. После выполнения данных операций значения регистров можно восстановить при помощи стека.

Команда LOOP

Одним из способов организации циклов в языке ассемблера NASM является использование ассемблерной команды LOOP.
Далее мы увидим, что циклы можно организовывать, например, средствами препроцессора.

```
1 mov ecx, 15
2 point1:
3   ; Ассемблерные команды
4   ; внутри цикла
5   loop point1
```

Для работы цикла необходимо использовать метку. Метка определяет место в тексте программы, куда будет передано управление на новой итерации цикла. Команда LOOP использует регистр ECX как счетчик итераций. Значение регистра ECX уменьшается автоматически, цикл завершается когда значение ECX достигает нуля.

Безусловный переход. Команда JMP

Метки в языке ассемблера используются не только для организации циклов, но и для организации ветвления в ходе исполнения программы, а также при определении функций.

```
1 jmp  m2
2
3 m1:
4   mov  ebx, 5
5   mul  ebx
6
7 m2:
8   add  eax, 10
9   sub  ebx, 20
```

Команда JMP реализует безусловный переход и является полным аналогом оператора goto языка C. В данном примере две строки после метки m1 будут пропущены. Для их выполнения, в программе должен быть другой переход к метке m1.

Условные переходы. Команды вида Jcc

Часто бывают полезны переходы, зависящие от условий. Перед командой условного перехода может стоять команда сравнения CMP или другая арифметическая команда. Переход к той или иной части программы зависит от результата выполнения этой команды. Например, JZ m1 означает перейти к метке m1, если результат предыдущей операции равен нулю (jump if zero)

```
1 cmp eax, ebx
2 jl func1
```

JL означает перейти, если меньше. Т.е. переход к метке func1 будет выполнен если EAX меньше EBX.

```
1 cmp eax, 10
2 jg func2
```

JG означает перейти, если больше, а JE - переход в случае равенства.

Команды вида Jcc

Ниже приведены условия выполнения основных операций перехода с условием.

JE/JZ	jump if equal/ jump if equal to 0	ZF = 1
JNE/JNZ	jump if not equal/ jump if not 0	ZF = 0
JC	jump if carry	CF = 1
JNC	jump if no carry	CF = 0
JO	jump if overflow	OF = 1
JNO	jump if no overflow	OF = 0
JS	jump if sign negative	SF = 1
JNS	jump if nonnegative sign	SF = 0
JP/JPE	jump if parity even	PF = 1
JNP/JPO	jump if parity odd	PF = 0