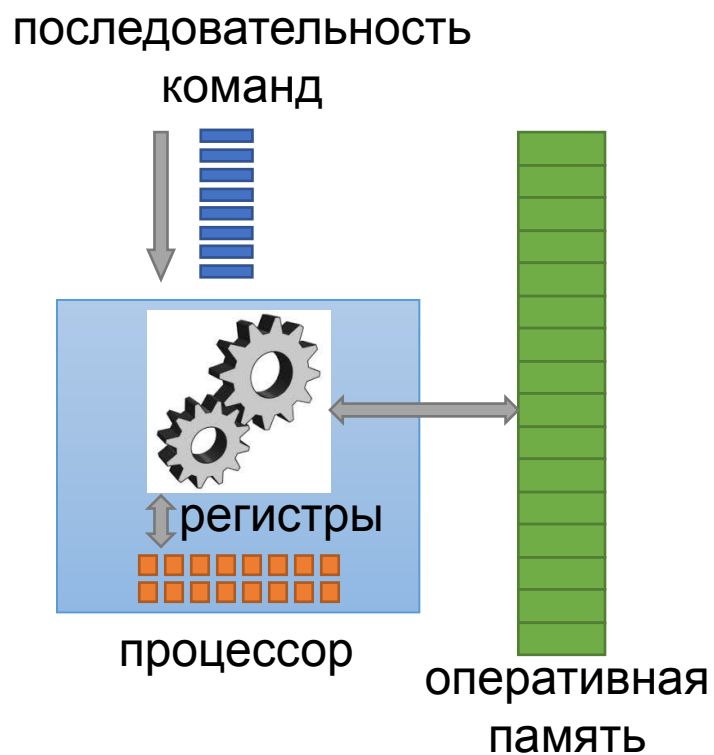


Краткий обзор языка ассемблера архитектуры x86/x86-64

Инструменты

- Получение ассемблерного листинга
 - `gcc -S prog.c → prog.s`
 - `icc -S prog.c → prog.s`
- Компиляция ассемблерного листинга
 - `gcc prog.s → a.out`
- Дизассемблирование объектного кода
 - `objdump -d a.out >prog.txt → prog.txt`
- Полезный ресурс:
 - Compiler Explorer, <https://godbolt.org/>

Упрощённая схема процессора



Регистры

- регистры общего назначения
 - имеют размер машинного слова (размер адреса в данной арх.)
 - некоторые части регистров тоже являются регистрами
 - количество регистров зависит от версии архитектуры: 8 / 16
 - используются для целочисленной арифметики и адресации памяти
- регистры сопроцессора
 - имеют фиксированный размер: 10 байт (80 бит) и количество: 8
 - используются для хранения вещественных 10-байтных значений
 - организованы в круговой стек, имена регистров задают смещение относительно вершины стека: `st(0)`, `st(1)`, `st(2)`, ...
 - в 32-битном режиме компилятор использует их для хранения вещественных значений,
 - в 64-битном режиме компилятор их использует, только если явно задана расширенная точность (`long double`).
- векторные регистры
 - имеют размер и количество в зависимости от версии архитектуры
 - могут хранить скалярные (одиночные) значения различных типов и вектора значений - столько, сколько входит в регистр
- флаговые регистры
 - 1-битовые регистры
 - хранят характеристики результата выполнения “последней” команды (определённые команды модифицируют определённые флаги)

Регистры общего назначения

32 бита

	16 бит	
	8 бит	
EAX	AX	
	AH	AL
EBX	BX	
	BH	BL
ECX	CX	
	CH	CL
EDX	DX	
	DH	DL
ESI	SI	
EDI	DI	
EBP	BP	
ESP	SP	

x86 32-bit

64 бита

RAX	EAX
RBX	EBX
RCX	ECX
RDX	EDX
RSI	ESI
RDI	EDI
RBP	EBP
RSP	ESP
R8	
R9	
R10	
R11	
R12	
R13	
R14	
R15	

x86-64

64 бита

	32 бита	
	16 бит	
	8 бит	
RAX	EAX	
	AX	
	AH	AL

Регистры сопроцессора

Физические номера	80 бит			Относительные номера
	1 бит Знак	15 бит Порядок	64 бит Мантисса	
0			mm0	ST(5)
1			mm1	ST(6)
2			mm2	ST(7)
3			mm3	ST(0)
4			mm4	ST(1)
5			mm5	ST(2)
6			mm6	ST(3)
7			mm7	ST(4)

Регистры MMX / 3DNow!

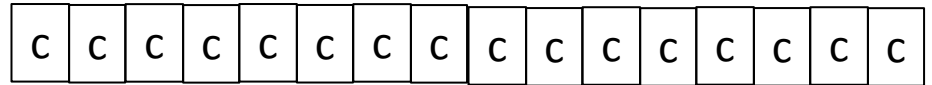
Векторные регистры

SSE

128 бит

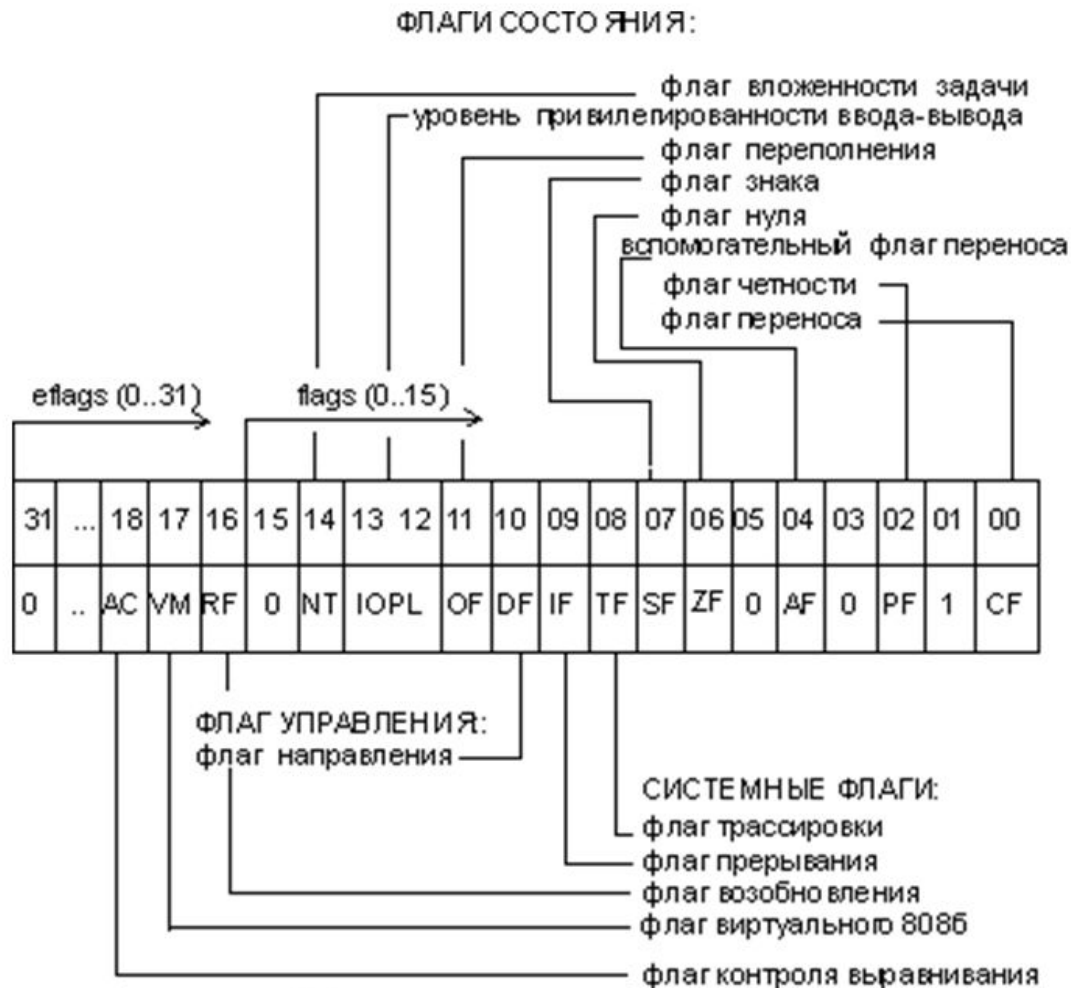
XMM0
XMM1
XMM2
XMM3
XMM4
XMM5
XMM6
XMM7
XMM8
XMM9
XMM10
XMM11
XMM12
XMM13
XMM14
XMM15

x86
32-bit



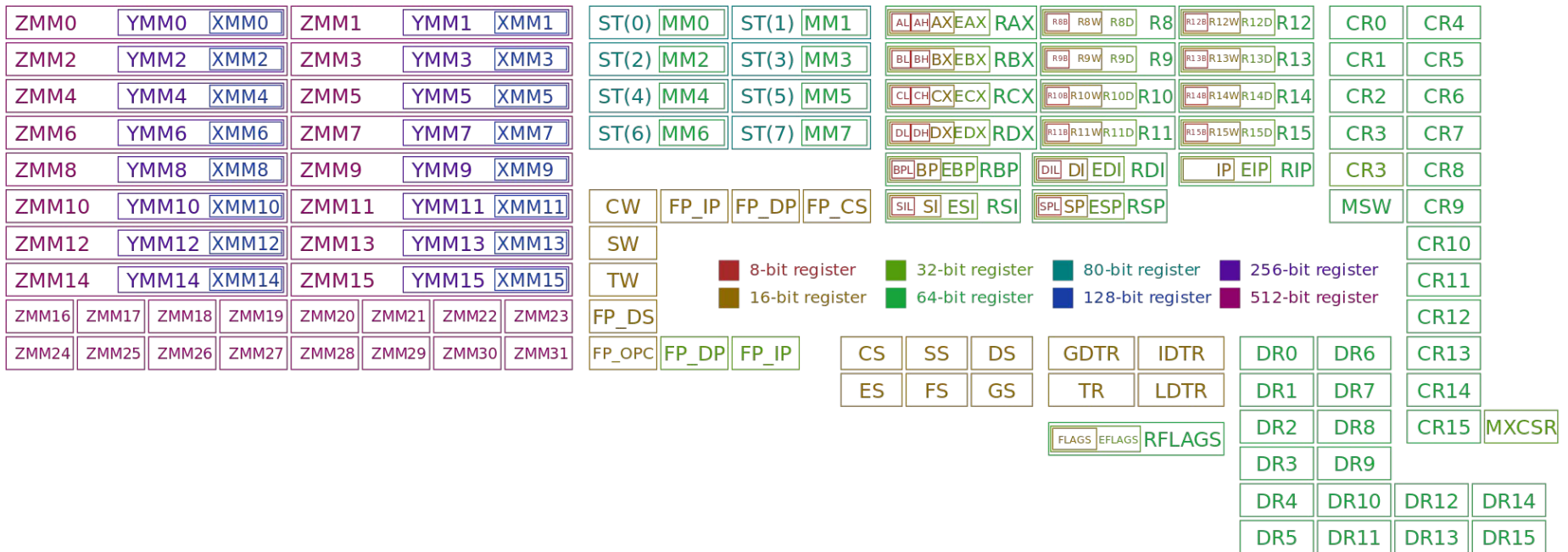
XMM: 16 байт
YMM: 32 байта
ZMM: 64 байта

Флаговые регистры (флаги)



Содержимое регистра eflags

Регистры x86



Адресация памяти

- задание адресного выражения:
 - Общий вид: $d(a,b,c) = d + a + b * c$
 - Сокращённый вид: $(a), d(a), (a,b), (,b,c), \dots$
- задание метки:
 - Например: `.LC1`
- неявное задание адреса
 - Например: `movsb → [rdi++] := [rsi++]`;

Команды

Формат команды:

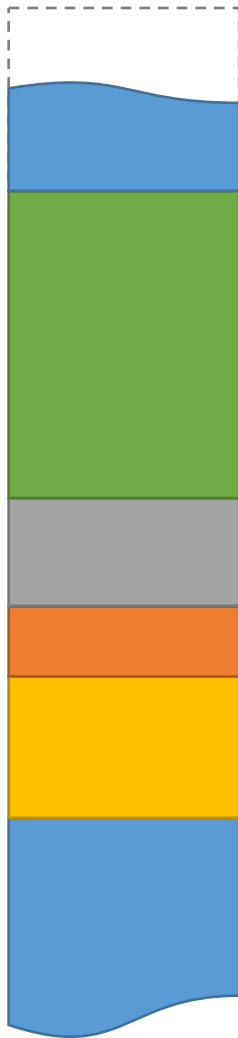
- Префикс
 - rep команда → while (rcx--) команда;
 - lock команда
 - ...
- имя команды – какую операцию выполнять
 - add, mov, cmp, ...
- суффикс - тип данных
 - b, w, l, q, sd, ps, ...
- параметры команды:
 - константы, регистры, адресные выражения

ОСНОВНЫЕ ВИДЫ КОМАНД

- копирование: mov, xchg, ...
- арифметико-логические:
 - арифметические: add, sub, mul, div, sqrt, ..., lea, ...
 - логические побитовые: and, or, not, xor, ...
 - сравнения: test, cmp, ...
 - сопроцессор: fadd, fmul, ...
- передача управления:
 - jmp, j__ (e / ne / g / ge / l / le /...)
 - call, ret, ...
- работа со стеком: push, pop
- работа с флагами: cl_, st_, lahf, sahf, pushf, popf, ...
- ...

Строение стека

Адрес: 0



Локальные данные подпрограммы (кадр)

Сохранённые значения регистров

Адрес возврата из подпрограммы

Параметры подпрограммы

Стек