

Архитектура операционной системы

X86 asm & debug

Начало

- AT&T
- Intel
- СИМВОЛЫ:
 - латинские буквы, цифры
 - % \$ * . , _
 - () : <space> <tab>
- Команды:
 - процессора
 - ассемблера (начинаются с .)

Метки

- `label_has_colon_at_the_end`:
- `.` – метка текущего адреса
- `0-1[f|b]` – обращение к ближайшей метке
 - `f` вперед
 - `b` назад

Данные

- .byte – 1 байт
- .short — 2 байта;
- .long — 4 байта;
- .quad — 8 байт

Регистры общего назначения

- `%eax`: Accumulator register — аккумулятор, применяется для хранения результатов промежуточных вычислений.
- `%ebx`: Base register — базовый регистр, применяется для хранения адреса (указателя) на некоторый объект в памяти.
- `%ecx`: Counter register — счетчик, его неявно используют некоторые команды для организации циклов
- `%edx`: Data register — регистр данных, используется для хранения результатов промежуточных вычислений и ввода-вывода.

Регистры общего назначения (2)

- `%esp`: Stack pointer register — указатель стека. Содержит адрес вершины стека.
- `%ebp`: Base pointer register — указатель базы кадра стека (англ. stack frame). Предназначен для организации произвольного доступа к данным внутри стека.
- `%esi`: Source index register — индекс источника, в цепочечных операциях содержит указатель на текущий элемент-источник.
- `%edi`: Destination index register — индекс приёмника, в цепочечных операциях содержит указатель на текущий элемент-приёмник.

Сегментные регистры

- `%cs`: Code segment — описывает текущий сегмент кода.
- `%ds`: Data segment — описывает текущий сегмент данных.
- `%ss`: Stack segment — описывает текущий сегмент стека.
- `%es`: Extra segment — дополнительный сегмент, используется неявно в строковых командах как сегмент-получатель.
- `%fs`: F segment — дополнительный сегментный регистр без специального назначения.
- `%gs`: G segment — дополнительный сегментный регистр без специального назначения.

Разные (но полезные) регистры

- `eip` – указатель текущей команды
- `eflags`
 - `cf` - перенос
 - `zf` – ноль
 - `of` – переполнение
 - `df` – направление в строковых операциях

Команды

- CMD source, destination
- Без операндов:
 - “чистый” синтаксис команд процессора
- С операндами:
 - суффиксы размера операндов: b,w,l,q;

Наборы инструкций

- General purpose
- FPU
- MMX technology
- SSE,2,3,E3,4 extensions (streaming SIMD)
- AVX extensions
- System instructions
- IA-32e mode: 64-bit mode instructions
- VMX instructions
- AESNI and PCLMULQDQ (кодирование и криптография)

Операнды

- \$ конкретное значение – \$0x732
- % регистр – %eax
- адрес

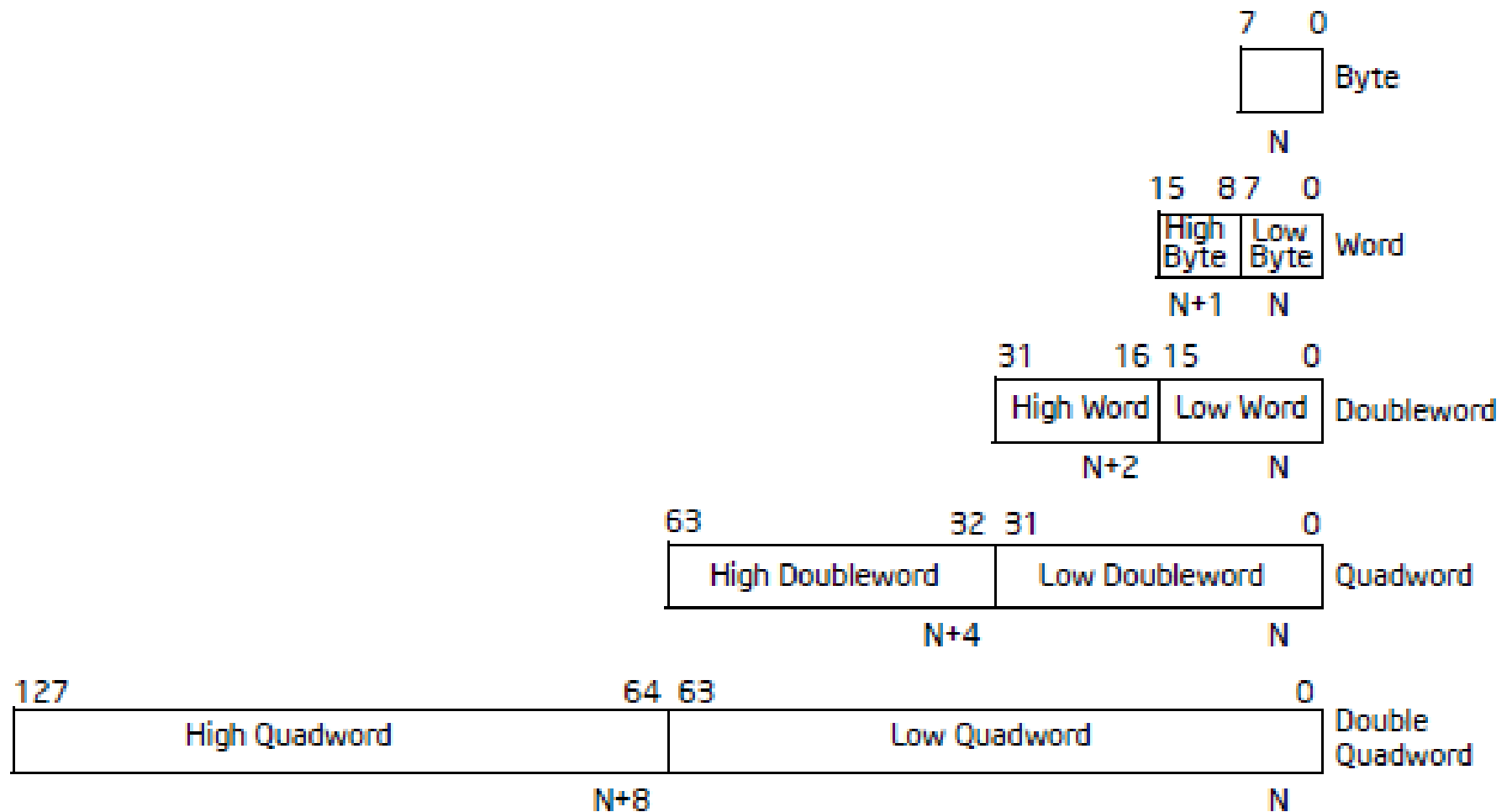
Важно: команды не могут использовать более одного операнда в памяти!

Секции

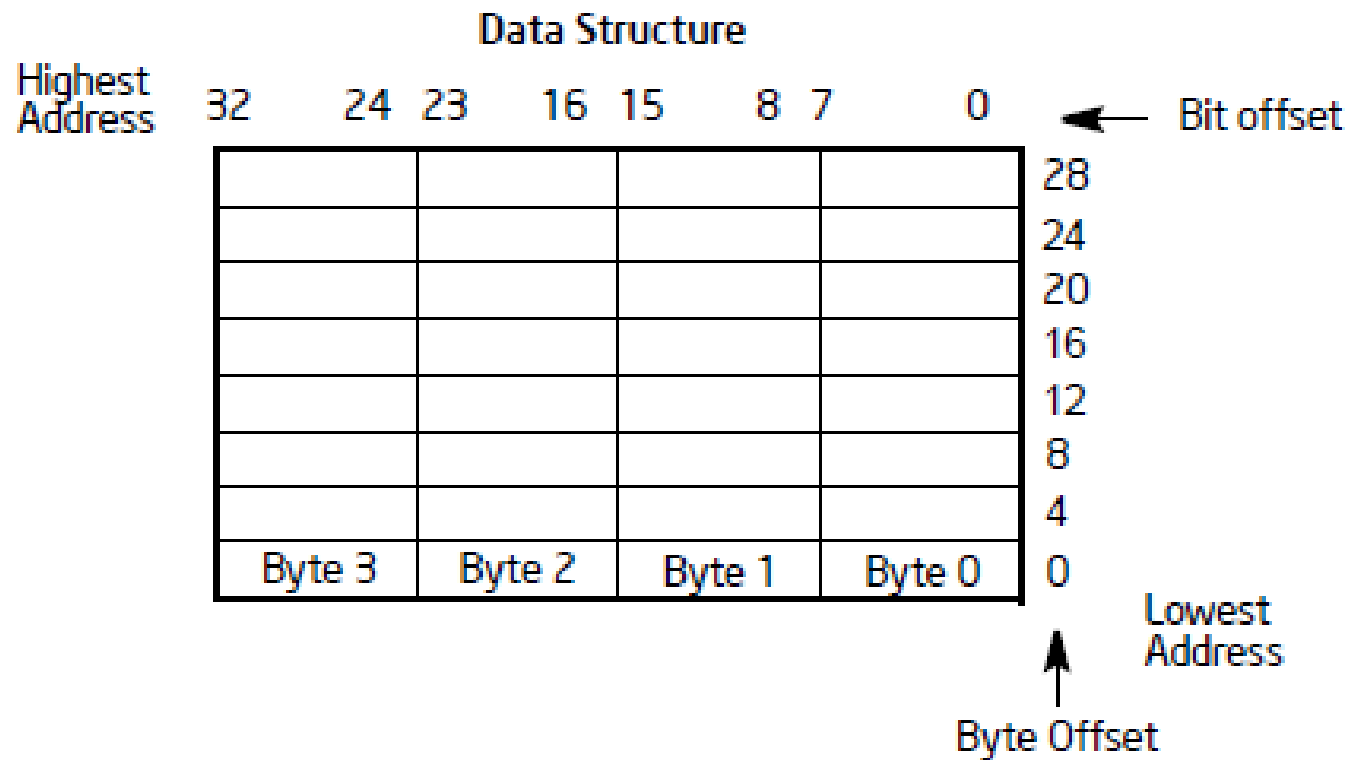
- `.text` – код программы
- `.data` – данные
- `.bss*` – статическая память
(неинициализированные данные)

*) You may allocate address space in the `.bss` section, but you may not dictate data to load into it before your program executes

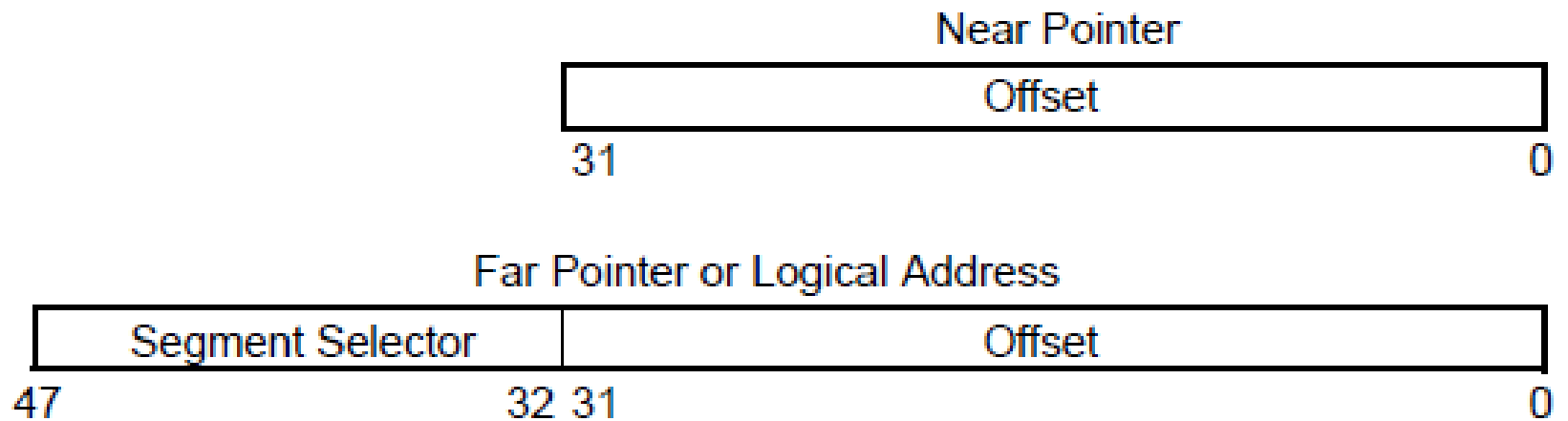
Фундаментальные типы данных



Структура данных в памяти



Указатели



Адреса

`offset(base,index,multiplier)`

- `multiplier = 1,2,4,8`

`address = base+index*multiplier+offset`

- Примеры:

- `(%ebx)` – адрес лежит в регистре `ebx`

- `4(%ebx), -4(%ebx)` – адреса по смещению `+/-4` от того, что лежит в `ebx`

- `some_string(,%ebx,4)`

Вызов функций

- call label
- передача параметров
 - в регистрах
 - в памяти
 - в стеке

Gcc keys for debug info

- `gcc -g<0,1,2,3>` – Debug information levels
- `gcc -o<0,1,2,3>` – Optimization levels
- `gcc -ggdb` – Produce debugging information for use by GDB. This means to use the most expressive format available

Starting gdb

- `gdb ./executable`
- `gdb ./executable -c core`
- `gdb ./executable -pid process-id`

Core file (core dump)

- recorded state of the working memory of a computer program at a specific time
- `ulimit -c unlimited` – turn on creating core files

GDB commands/activities

- Getting information
- Execution
- Line execution
- Break and watch
- Stack inspection
- Sources navigation
- Printing variables

Getting information

- help <command>
- info
 - args
 - breakpoints
 - watchpoints
 - registers
 - threads
 - signals
- where

Execution

- r/run
- r/run arguments
- c/continue
- continue number – cont. ignore break
- finish – continue to the end of function
- kill
- q/quit

Line execution

- step (into a function)
- next (next line of code)
- until line-number
- stepi/nexti step for assembler instruction

Break and watch

- break function/line
- break + or – number of lines
- break filename:line or filename:function
- break *instruction_address
- break ... if condition
- break line thread tid
- clear
- enable/disable
- watch condition

Stack inspection

- `bt/backtrace`
- `f/frame [number]`
- `backtrace full`
- `up/down number`
- `info frame`