

Лекция Иван Бражникович

"Избыточные программы."

vanya.cpp@gmail.com

13.02.2014

[x86 8086]

Процессор работает с оперативной памятью хранящей в байтах.

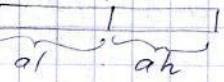
За раз обращение идет по 16 байтам

Существует регистр определяющий IP - instruction pointer. Процессор на основе IP читает из памяти IP, номера передаваемые IP из памяти

Byte	11111111
------	----------

Команда: ① MOV a b (a := b)

8 регистров GPR (general purpose register): ax bx cx dx si di sp bp - 16 бит

IP - регистр b 16 бит тоже
ax:  можно одновременно 2 байта

Байт MOV ax, 100 - 3 слова команды

B8	3	100
----	---	-----

(B8 - "MOV ax...")

MOV ax [bx]; (ax := bx) / сохранение байтами
MOV[bx] ax; (bx = ax) / изменение байтами

Endian (x86 - little endian - маленький байт впереди)

0x12	0x34
------	------

little endian последовательность 0x3412
big endian 0x1234

MOV Ax [bx]
MOV[ax] [bx] - не срабатывает

② ADD a, b ($a + b$) ③ SUB ab ($a - b$)
 ADD [ax] bx

④ MUL a ($a_x \cdot d_x = a \cdot a_x$) ⑤ IMUL (signed MUL)
 (unsigned) 16 bit

⑥ DIV a ($a_x = a_x : d_x / a$, 32bit/16bit)

$d_x = a_x : d_x \% a$
 однозначно
 uniquely

принцип
 (no hazard)
 генерирует
 мультипликатор, но не
 делит с остатком
 делит с остатком
 можно вправо и сам опред.

⑦ IDIV - signed DIV

Переполнение
 возникает при
 $0x8000 / -1$

⑧ NEG a = -a
 NEG ax; NEG [ax] word, byte

⑨, ⑩, ⑪ AND, OR, XOR a, b ($a_8 = b$,
 $a_1 = b, a^1 = b$)

⑫ NOR a, b ($a \sim b$)

⑬ SHL a, b ($a << b$) ⑭ SHR a, b ($a >> b$)

⑮ SAR a, b ($a^{arith} >> b$) ⑯ INC, DEC

SAR 77654321 u3 76543210
 арифм: SUR 07654321

⑰ JMP a (переход на метку a) 
 a:
 основные параметры

Инструкция / байтами:

CMP ax bx - имеет 8 регистров FLAGS
 (ZF, CF, OF, SF)

a бито 16 бит

Проверка на jz, jne, jc, jnc, ...

jz	equal	unsigned
jne	not equal	
ja	above	
jae	not less	
jb	below	signed
jbe	-11-	
jl	less	
jle	-11-	
jc	greater	signed
jge	-11-	

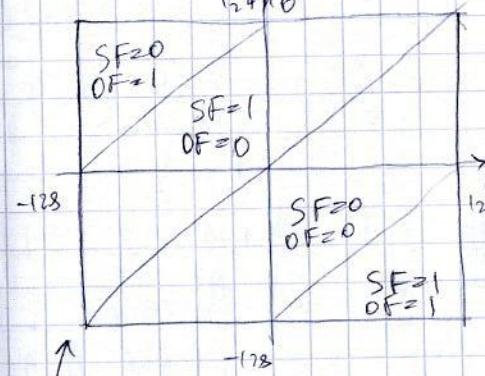
① ADD, SUB more
 Высоковыраженное для
 практик

ZF - zero (регистр = 0)
 CF - carry (заканчивается не 0)
 OF - overflow flag
 SF - sign - знак статического
 битов

⑤ CMP и SUB проверяют одинаково на
 запись практик, $a = b, a - b = 0$;

jz ~ jz
 jb ~ jc
 ja ~ cf == 0 & zf == 0

Разделение с signed: $a - b$



jge == SF == OF
 jl ~ SF != OF

Однако же более
 корректно

ZF=1
 OF=0
 SF=0

*OR ax ax
 MOV CX 10
 loop:
 add ax cx
 dec cx
 cmp cx
 jne loop

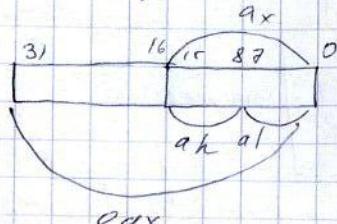
(XOR ax, ax - однозначно
 Intel запомнил ax, не надо
 MOV ax 0)

Easier way to do

Что такое биты слова для 16-битных регистров

Intel 386

$ax \rightarrow eax$
 $bx \rightarrow ebx$
 $cx \rightarrow ecx$
 $dx \rightarrow edx$
 \vdots

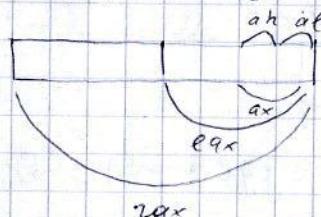


Слово называется lea : $ax + 5: lea ax, [ax + 4 * al]$

($0x00000000$
 $0x0000$)

200 ит
 appeal, временно
 имеет значение bx

А норма 8×64 $eax \rightarrow 2ax$, $ebx \rightarrow 7bx$...
 rieburkis esise & переведя 28 29 - 215.



20.01.2014 (Продолжение, переход к байтам)

Experiment: C/C 4.7.1

div 1: mov eax, ed;
 div 2: mov eax, ec;
 shr eax, 1;
 div 3: mov eax, ed;
 mov edx, -1431655765; = 0x00000001 > $2^{32}/3$
 nul edx;
 shr edx;
 mov eax, edx;

Деление занесение:

div 2: mov eax, ed;
 shr eax, 31;
 add eax, ed;
 sas eax, 1;

Смесь:

push reg/number / [in + reg + {1, 2, 4, 8} * reg];
 mov [sp - 2], a
 sub sp, 2

pop reg/number / [in + reg + {1, 2, 4, 8} * reg];
 add sp, 2
 mov a, [sp - 2]

Call label 1:

push next
 jmp label1

Ret N: pop trap
 add sp, N
 jmp trap

Example f(1, 2, 3) Push bp

push 1
 push 2
 push 3
 call f

Mov sp, bp

mov bp, sp
 sub sp, 100
 pop bp
 ret

- frame + frame - pointer

27.02.2014

Прерывание (interrupt) - 286 прерываний
модель x86

8-15 8086 прерывание (9 - штрафматура)

Анонс 8 - timer

0-7 проверки на "все хорошо"?

0 - сигнал на ядро (при детекции на 0 быт
использовалось interruption 0)

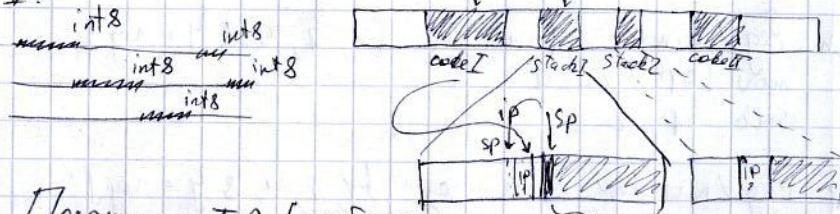
0x3F в Windows, 0x80 в Linux - это
прерывание

Процессор 0x80:

На x86 в когане памяти лежат массивы
из 256 ячеек памяти с адресами 4-бит
прерываний. В 0x80 хранение это другое.

Более сложные механизмы

Есть программы, в выполнении работают I,
если прерывание II, отрывою ^{IP} _{SP} отдает управл.

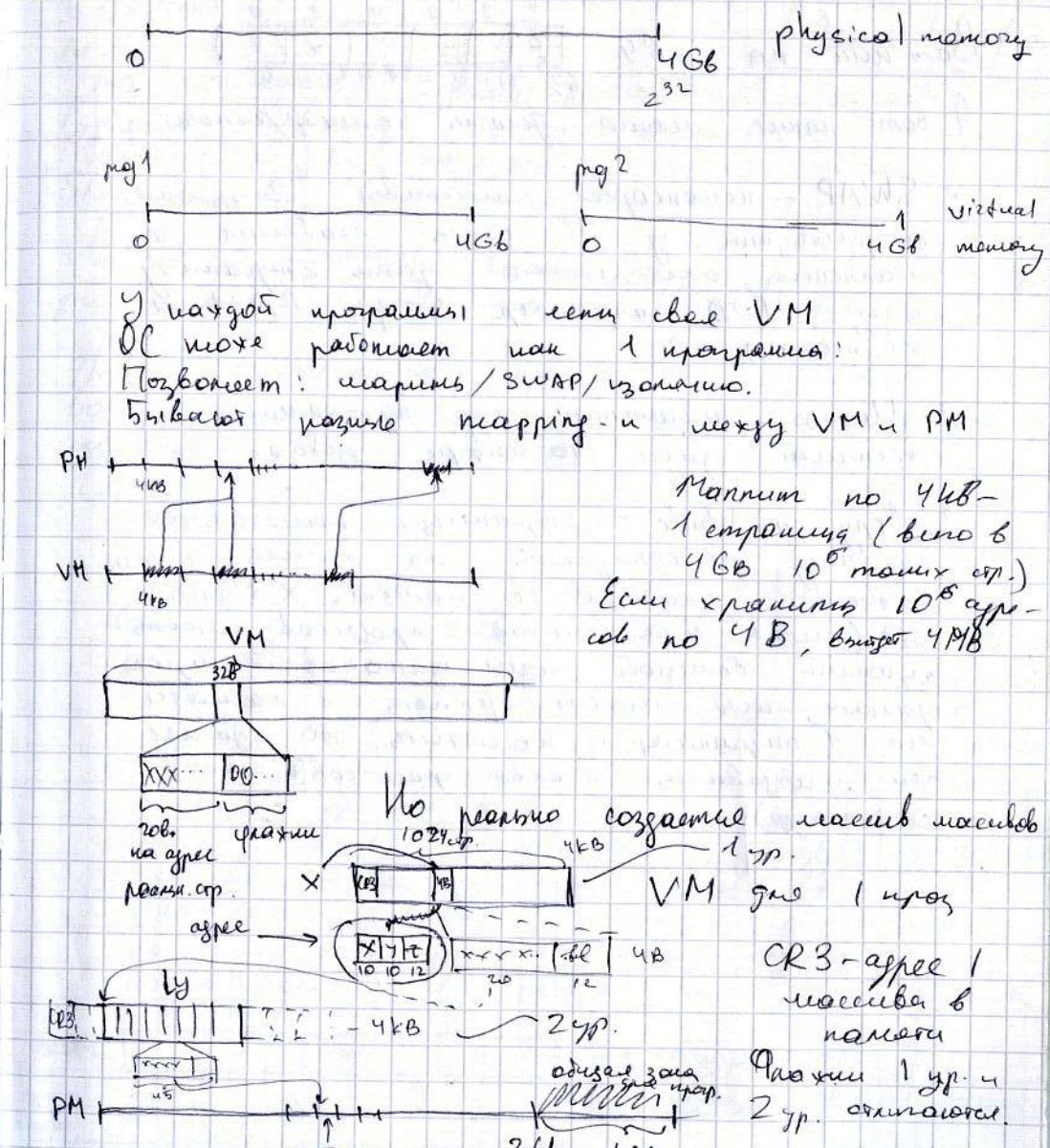


Помогает IP, (помощь
саму переписывает стек), и
указатель на верхнюю
стека, а там лежит другой IP. Ну охвачено.
IP&SP помимо переписывает между собой стеки
само по себе не запрещено

PROTECTED mode - режим, в котором программы
запускаются друг от друга. Все преры в системе имеют этот режим.

Механизм называется Paging (использование байт-кода)
(также есть CISC-расширение - страницы)

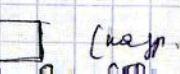
8086 32 bit - первый механизм ядр. с Paging.
может приводить до 4 ГБ



Массивов управление 2 способами, можно выбрать
(выделение по 1 массиву или по нескольким 4 МБ в
единицах (РМ))

Система замечательна, это то

x64 все таки:



Вам надо на $\times 64$:

А вам надо можно сделать следующим:

1. SWAP - неподобные массивы 2 уровня осуществляют, что ОС сеанс памяти разделяет, они могут занимать страницу в RAM, например (если RAM не хватает)
2. Можно отмечать что программы не касаются друг друга. Это иначе удобно
3. Copy-on-Write - страница памяти страницу подсчитывают на записи, а при записи меняется ее значение. Х3 записи это (если сеанс много процессов, память делилась когда изменяется адрес которых, или может удаляется в памяти то 1 единицер и нолью; это удобнее, чем изграждание [когда процессов больше] страниц).

6.03.2014

М

мои источники: Бубнов "Ассемблер"
Руданов Романов "Линии ассемблера"
"Plucker's delight" (на русском мало переведено)
"Shen Lipasti" Modern microprocessor design

Лекция по мониторам "Ulrich Drepper: what every programmer should know about memory"

glibc - библиотека для С от GNU.

bottleneck - то, во что упирается холода канала передачи (расчет pipeline)

1. Кэши;

Объясняет принцип создания кэша.

Кэш-память $\sim 64 \text{ B}^6$ L1/L2/L3 $\sim 20\text{ Гб}$
Как работает:

Процессор обращается к памяти по адресу, $I32B$,
то есть Хэн-память массив из N блоков,
в каждом фиксировано список места (L1, L2, ...)
(блок, значение); Обращение $I32B \rightarrow (L1, ..., L2, ..., L3)$ (если, если у нас Хэн-память)
(то где в находятся блоки фиксированы
и не-пар-это ассоциативность
имя (один и тот же адрес памяти)).

ROM:
номер блока

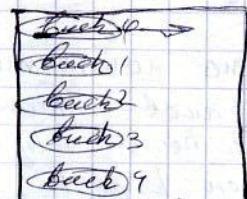
Это надо залог в памяти, надо есть
хэшинга) страниц по Hash) фиксировано и не делают
ее вычисляем и не делают по Value,
но есть биты Value из этого.
Хэн-память хранится в SRAM быстрой;

В связи с тем что спецификация
создана:

Банко!!!

Такое, например:

```
N = 2k; float a[N][N];  
for (size_t, i=0; i< N; ++i)  
    for (size_t, j=0, j< N, ++j)
```



Если наше

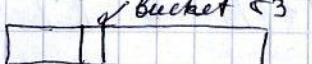
no $i \geq l \geq j$, no
и back \leq back0
то багажник.

A $\}$ $a[i][j]$ something
example

$a[j][i]$ something

А если мы будем бояться багажника swap ($a[i][j]$, $a[j][i]$), то с $2^k = N$
также есть $2^k = N \pm 1$ из-за блоков
в первом блоке равно по времени. В первом
случае ($N = 2^k$)

Быстро ищем в кэшах адреса



If = address % 100

	data1	53
0	data2	163
1	data3	253
2	data4	653

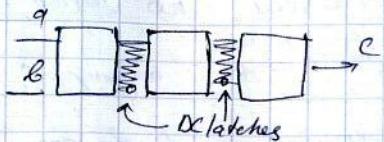
напишем так.

2. Pipelining

Лучше у нас есть схема, где времена $c = a + b$
затем b и a
затем c

Но это работает гораздо лучше, потому что можно
менять последовательность последовательности

DC-latch (DC flip-flop)



Так работают при
pipeline, потому что

RISC делают попытку, то он хорошо
использует на простые арифметические, такие
MIPS (автоматическое использование попыток
попытка — [root] [load] [write] — несет больше рисков для
багажников ~~Structural hazard~~)

Hazards:

structural, branch prediction, speculative execution;
branches, kill flags,
call ret; ? Google it;

А также очень важно не делать старт

===== * * *

13.03.2014

C / ... / C++

Лекция вторая про разнотипы.

Types: unsigned / signed

short, int, long, long long

Также unsigned long a;

a ~ signed a но geography

Первый Unix написан на C, но никаких бинарных типов, например портируемых из языка C и его же в 80-х было мало, поэтому для разного написания, например, функций, используется фокус:

POSIX - unix, ANSI C - C языковые

Системный разработчик - ABI, однородные идентичные функции (такие программы должны работать одинаково, так как описывают функции в т.ч.), ABI зависит от архитектуры и языка.

ABI:

	Microsoft		GNU		Titanium
32		32		64	
short	2	2	2	2	
int	4	4	4	4	
long	4	4	4	8	
long long	8	8	8	8	
char	1	1	1	1	
Signed char					
unsigned char					
float (single)					
double (double)					
long double (ext)					

— // —

Single precision:

of: 0/0000000/0...0 } 2f: 0/10000001/0...0
lf: 0/10000000/0...0 }

Struct:

struct point3{

point x
double y

struct
point {

int x
int y

};

Места: sub esp 4

mov [esp], 4

call g()

mov eax, [esp]

add eax, eax

mov [esp], eax

add esp, 4

ret

int f()

{

int b=0;

g();

b+=2;

y

(cdecl)

Функция имеет - calling convention, — соглашение о том, как передавать аргументы в ABI одинаково, то есть 8 байт на каждый параметр:

void f (int, int, int)

В cdecl все в одинаковом порядке:

push 3

push 2

push 1

call f

add esp, 12

3 int given

f(1, 2, 3)

→ Взять за Var args...
указать булево значение

сравнение: f(x, ...)
(void printf(char const*, ...);

Но меньшее значение, зная наименование разряда `sub esp, 12`

```
-- stdcall  
push 3;  
push 2;  
push 1;  
call f  
f: ret 12
```

А если если -- fastcall,
который передает 20-го байта
регистров.

Что это значит разряд calling conv,
или что это на assembly?

Документация: <http://sonokin.github.com/>

Также `helloworld` в `obj`, что это
значит unsigned int и `sub`.
Код заморен `NASM x86-64 Linux`

20.03.2014

Большинство языков assembly не DOS:
DOS

```
0x21  
ax,bx,cx,dx  
MASM/TASM  
segment.text  
x86
```

```
syscall  
rax,rbx,rcx,rdx  
NASM  
section .text  
x86
```

Чтожем:

```
mov ebx,...  
mov eax,[ebx]
```

в assemblyе указание байт.
в C++ указ. типов

В C++: `int n - число, int *n - указатель`

`a = &n` — а получает ссылку на `n`
`int m = *a` — получает число по ссылке.

Можно пасовать указателями:

`short *b = (short *) a;` — то же самое что и `b = a`,
но можно не звать `reinterpret_cast`, просто, лучше.
Так же можно пасовать `char`.

Есть указатели на void:

`void *c = (void *) a;` — оно различимо, because

it можно так:

`int **b = &a, if a = &b, if b = b, например`
`int **b = &&n. не багио:`

`a = b` value = value

`b = b` не багио, b - value

`(a + b) = c` тоже не багио

`&a -` value, `a` — контекстуально `b` value

`int **b = &(a = &n)` багио, `++++a` тоже, `a++` тоже

С указанием нуля в кономах (xa)

Массивы:

int a[100] - массив, адрес a[i] = i[a]
int b[100][100] (b[i][j]: 0 ≤ i < 100, 0 ≤ j < 100)
Массивы можно использовать в указателе
int * c = a; // c[0] == a[0], *c == a[1]
Наг. указатели определены операции `new`-
`delete` / выделение памяти (`++` тоже)
int d = a[5]; // *d = a[5].
* becomes `new` с помощью `new`.
`int * a = new int[10]`

Применение использования в качестве имени разные
массива переменную не с `new` можно в
C++14 и C99

int * a[100] - массив указателей
`int(* a)[100]` - наз. не массив.

int *** (**** a [10][20][30]) [40][50][60]
 ^ ^ ^
 1 2 3
 | | |
 5 4 6

$\star\star\star(\star\star\star a[1][2][3])[4][5][6] = 10;$

Указатели на функции

mov ebx,... } void f(int) { } int main() {
: call ebx,... } } g = &f; void (*g)(int) = &f;
call ebx,... } g(10); }
 ^
 1
 bx

(Quake 3): float Q_sqrt(float number) {
long i; float r2, y;
const float threehalfs = 1.5F
x2 = number + 0.5F

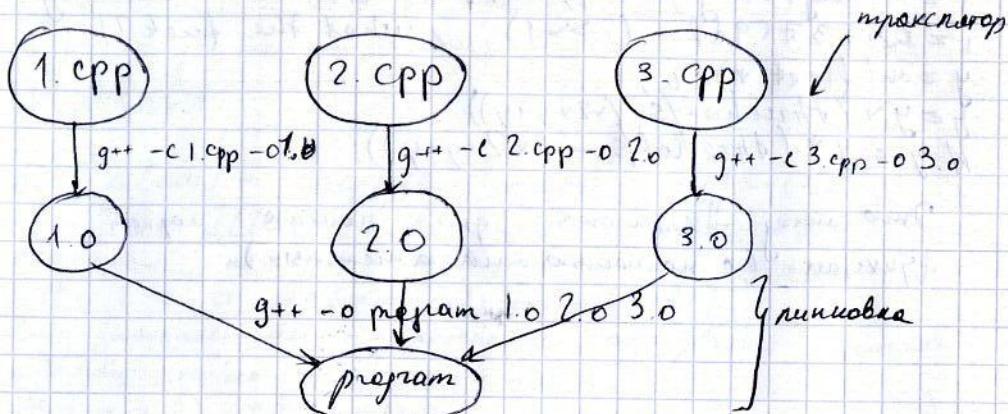
y = number

i = *(long *) &y; // evil point bit level hacking
i = 0x5f3759df - (i >> 1); // what the fuck?!
y = *(float *) &i;
y = y * (threehalfs / (x2 + y + y));
y = y * (threehalfs - (x2 + y + y))

Это mere фокусы где наше право
пользования (с помощью указателя)

23.03.2014

Личное



А если можешь скомпилировать это в g++ -O program
1.cpp 2.cpp 3.cpp. Минимальные изменения генерируются,
также отдельно для каждого языка. Для каждого языка,
cpp → 2.0 assembler → assem. o (разные языки), а
потом скомпилировать. А если можешь
сделать о. файл можно безо
проблем.

global foo; foo: mov eax 0x... ...	extern foo; call foo;
--	--------------------------

B.cpp: int main() { } - yxe global

→ void foo(); - extern foo

declare > static void bar() - private

define

Function may be declared lots of time but only once - defined.

Градиентное определение:

int a; - definition

static float b; — definition (private)
extern int a; — declaration

repercussion

22

Фундамент

```
extern int a;  
int a
```

declaration
definition

```
void f();  
void f() { ... }
```

Структурни и компоненти на водите неизвестни по
именование, лице и использован;

Процессор

the common gene

Метод наследования
g++ -E 1.cpp
-S -ассемблер

```

graph LR
    A((1.epp)) --> B((preprocessing))
    A((1.epp)) --> C((1.i))
    D((1.0)) --> B((preprocessing))

```

```
Программист не имеет на это право,  
а #include "a.h" // a.h  
int main() { var(); } 3  
f()  
3
```

а склоняется развернутая в глагол.
Если есть `#define X 100` — макрос, который
меняет все "X" на "100"

#define X Y
#define Y X XY → YX (X → Y → X Y → X → Y)

Перенесите оба вида α в левую часть уравнения: $\# \text{define } SGN(a) \quad ((a)*(a))$

if def M (seen def m, unknown) Then if (comes true)
From user

30.04.2014

Константы

#define PI 3.1415

Но так скомпилирует, что-нибудь, может быть
указанные значения int PI

double const PI = 3.1415;

Такое тоже: мы не можем менять значение.
const, неко и int const * = &PI

int a;
int const b;
int * a;
int const * b;

a = b
b = a // error a = b // error, int * type, const required
b = a

int * a;
int * const b = a;

Foo: const int a[5] ~ int const a[5]

Структ: struct

2	x p	p. a = 5 // ok
int a;	x const a.	p. b = 5 // error
int const b;		g. a = 5 // error
3		g. b = 5 // double error

const casting.

int * p = (int *) &a // ошибка, надо UB

* # *

Inline-функции

Можно сделать #define max(a,b) ((a)>(b)) ? (a) : (b)
Но! max(a++, b) — ошибка

Можно сделать такую функцию:

11. h

int max(int a, int b)

11. main.cpp

#include "1.h"

int main()

max(5,6);

3.

Но если это включаем, то все ошибки:
LTO, LTCG, но все окей

11. main.cpp

#include 1.h

int main()

max(5,6)

3.

11. 2.cpp

#include 1.h

11. h

int max (int a, int b){
 return a > b ? a : b;

3.

Синтаксис не ограничен
Если использовать макрос -O2, то он будет
распознавать реальное значение, а не значение #define. Но
11. h inline int max() ... — всегда ошибка (если)

Конструирование

Конструкторы:

struct complex

{

float re;

float im;

3.

запись
значения
факт

Что-то надо записать

struct complex {

complex() {

re = 0;

im = 0;

}

float re;

float im;

complex(float re, float im)

}

this → re = re

this → im = im

y

num → complex(1, 2),
наименование

new complex(1, 2);

создана на стр.

10.04.2014

Лекция, уроки

struct mytype

{

mytype();

mytype(int, int);

~mytype();

y

void f()

{

mytype a;
g();

y ← вызовение mytype a

Несколько параметров:

Демонстрация багажа в виде compound statement ({ ... }) — при выходе из него (return / break / continue / throw) разрушение буферов не происходит непосредственно.

void f(mytype, mytype)

void g()

{

f(mytype1); mytype(2, 3);

:

y

**.h
struct mytype2

{

mytype a;

mytype b;

mytype 2();

~y

mytype2()

y

**.cpp
mytype2 : mytype2();

 { a (1, 2)
 ; b (3, 4) }

~y

Т.е. это то же самое.

В этом случае
одинаковый результат
имеет место в обоих случаях.

Причины различий —
это различные goto
безопасные main, а неодинаковые
main один из которых.

References

int * a; — указатель на int

int & a — ссылка на int

указ.

```
int a;
int * b = &a // int * b = int&
f(*b);
++*b
g(b)
b = ...;
```

const

```
int a;
int & b = a;
f(b)
++b
g(&b)
N/A
```

Пример неоднозначности

struct big-integer

```
{  
    big-integer();  
    ~big-integer();  
    big-integer(big-integer const& other);  
};
```

```
big-integer a;  
big-integer b = a; ~big-integer b(a)
```

a[10] ~ int& operator[] (size_t i)

a[10] = 5

Значит это же указатель:

struct array

```
{  
    array();  
    array(array const&); // default ~ array a;  
    array(const&); // array(b); array b = a  
    array();  
    array& operator=(array const&) // b = a  
    ! struct integer {
```

// Пример - integer operator+(integer const& a,
// integer const& b)

// Краткий пример:

```
int * a;
int + const b;
int const + c
int const * const d;
int const int * const e;
const int + const f;
const int const * g;
```

} б/c неоднозначно, non.

г) const bool z:

```
int & a;
int const & c;
```

integer & operator+=(integer const&)
integer & operator-=(integer const&)
-a = ... - разные
a += b - ошибка

integer & operator++(integer &)
integer & operator++(const integer const&, int) ++a
bool operator<(mytype)
struct mytype

size_t width;
size_t height;
y.

bool operator<(mytype const& a, mytype const& b)

```
{  
    return a.width < b.width &&  
           a.height < b.height;
```

(too many comparisons ! (a < b) & ! (b < a) >
a == b,

17.04.2014

1. h

static int a = 5; - будет выведено 5 при компиляции программы
static - нужно помнить в генераторе единого определения.

inline - функции - несколько одинаковых (genus), Sintax ошибки

gcc - компилятор

ld - компоновщик

gold - отладчик компоновщика от компании ld

gold поддерживает только ELF, явно не указав
наименование ld, но работает корректно
даже если.

1. h

static void f()

- будет выведено в качестве параметра
но в компиляторе MS или
с gold неизвестное
исключение ICF.

ICF срабатывает единовременно

Incomplete types

```
/* a.h */  
#ifndef A_H  
#define A_H  
#include "b.h"  
struct a {  
    struct b  
};  
b * bb;  
y;  
#endif
```

```
/* b.h */  
#ifndef B_H  
#define B_H  
#include "a.h"  
struct b {  
    struct a  
};  
a * aa;  
y;  
#endif
```

В чем проблема?
1) первоначально включена
a.h., ранее b.h
2) единовременное
a > bb; - not defined

Зависит #include "b.h"
но "struct y;" never
used

Обработка мусора
с помощью forward-declaration

уходит, ведь при изменении 2. h, где есть a из
1. h, зависимости не будут пересчитываться.

Бог знает [struct B;] B - incomplete type;

Анонимное оно так, что нужно пересчитывать
все зависимости, применяются новые
сравнения данных.

Число звездочек incomplete type:

struct *

x * a //ok

* b //error

No errors

struct * z y;

a * h(); //no

a -> f(); //error

a . g(); //error

No errors

3 звездочки не incomplete type

Использование free параметра символов памяти

void f()

{

int a[10]

a[20] = 7;

y

malloc leak,

(например срабатывает non-alloc

malloc - free)

Analyzer - dr. memory. (анализатор памяти)

Системы памяти - sanitizers - address-sanitizers
встроенные в gcc, проверка памяти valgrind
sanitizers обнаруживает ошибки!

Econs uses pageheap.

24.04.2014

Inheritance

struct base

{ int a;

};

struct derived::base

{ int b;

};

derived d;

d.a = 5;

d.b = 6;

base & b = d;

b.a = 7;

base + b = & d;

b -> a = 8;

Конфликтов не находим

base (int a, int b) { ... };

derived () : base(1, 2);

Было одно право выбора наследства 4-15.

base { derived

void f()

{ }

derived

void f()

{ }

derived d;

d.f(); // b

base & b = d

b.f(); // a

d.base::f(); // a

};

y

base { derived

virtual void f()

{ }

derived

void f()

{ }

y

Применение:

- 1) struct base { void f() { ... } }
- 2) struct base { base f() { ... } }

Применение не находит идентификатора f(), а нормально его ищет как наследование.

Но можно это out-of-class-declaration!

- 2) ?? however simpler

abstract: virtual void log(string const& a) = 0;

Но не можем создавать объекты этого

класса (некорректное использование)

struct x { struct y { struct z

int a;

int b;

};

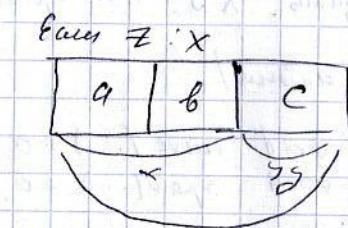
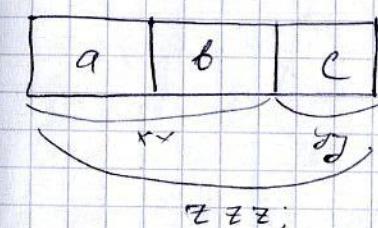
float c;

};

x xx

y yy

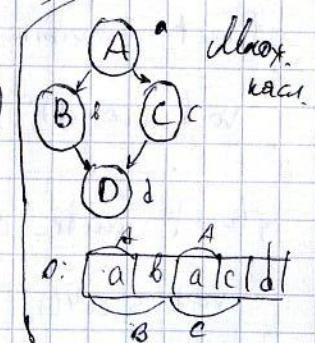
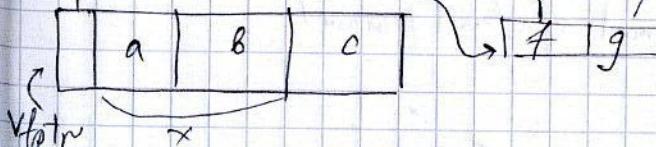
z zz



x { virtual void f(); }

y { void f(); }

z { void f(); }



struct base1 { int a; f() }

struct base2 { int a; f() }

struct derived : base1, base2 { }

derived d ; d.base1::a ;

d.base2::f(4)

А кеңең көрсеткіштегінде
(міндеттес көрсеткіштегінде)

struct A

{
};

struct B: virtual A

{
};

struct C: virtual A

{
};

struct D: B, C

{
};

Нан зеңдең? X3. Бар ныңыз:

struct animal

{
};

virtual void move(..) = 0;
virtual void speak(..) = 0;

3.

struct swimming_animal: virtual animal

{
};

void move();

3;

struct mute_animal: virtual animal

{
};

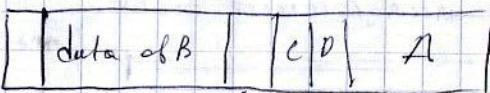
void speak();

3;

struct fish: swimming_animal, mute_animal

{
};

D d;
d, a; // Banqao
A&a = d;



v6ptr v6tu

08.05.2014

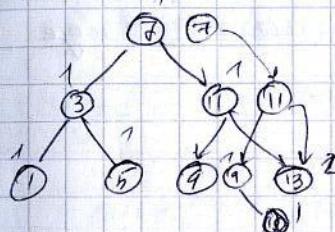
square ~ rectangle

Квадрат ~ квадрат от квадрат.

1. Квадрат номинальный
2. Квадрат

A besides represent order: y aux only. Stack-based
copy-on-write

// name; google Btrfs



Small object optimization — мемориалдың
оңайлық барынан
Overload resolution
Copy
relob

У нас не будет shown на запомощен
использование. Установка в зоне атаки
— SEH or Microsoft — access violation.

How to write basic code:

1. Некий компонент создает память
все изменения неприменимы для try.

А как в случае с исключением?

А какого рода все не применимы?

(NB) Если exc в конструкторе, то оно не
использовано.

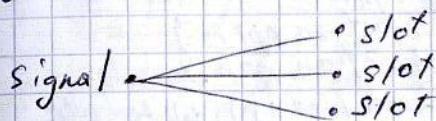
В swap'ите неожиданно бросит except.
swap-trick
1 copy(obj)
2 (copy)
swap(.., ..)
3 string & operator(string const &
rhs);
string copy(rhs);
swap(copy, *this);
return *this;

22.05.2014. Прототип исключений (Exc2)

UI - user interface

Приведем возможные способы обработки исключений

signals / listeners / observers



signal emit —
однократное сообщение.
слоты.

struct mytype

2

mytype (model&m) : m (m)

exc here

m. object_created(). connect(.. this),
m. object_destroyed(). connect(.. this),

3

~mytype()

4

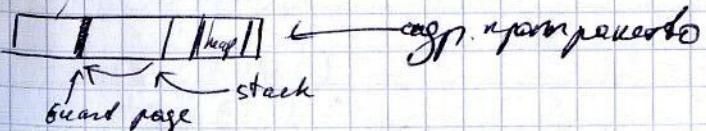
m. object_destroyed. disconnect(this),
m. object_created. disconnect(this);

5
private:

model&m;

6

model и user



new биржеем б хунд, дз new б сюне

RAII:

new
new / exc

del

del - most deleted

Smart
pointers.

struct unique_ptr

```
unique_ptr() : ptn() {  
    unique_ptr(unique_ptr<const T> p) = delete;  
    unique_ptr & operator=(unique_ptr<const T> p);  
    explicit unique_ptr(int * p) : ptn(p) {}  
    unique_ptr& ~unique_ptr() { delete ptn; }  
    int * operator*() const { return ptn; }  
    int * operator->() const { return ptn; }  
private:  
    int * ptn;  
    void reset(int * p) { delete ptn; }  
    int * release() { ptn = p; }  
    int * ptn;  
    delete ptn;  
    ptn = 0;  
    return p;
```

C++ in-depth

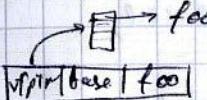
Сиот Марк
Герб Камп
Андре Аланчукевич
> Борис Чиркович
Effective C++
C++ gotchas
Modern C++ design
C++ programming language
Principles and practices

C++ object model - no Layout

19.05.2014

Слово unique.

- Что такое C++.unique?
- Это такое название, которое говорит, что f не имеет базового понимания.
- Слово unique, потому что оно используется, например в fpn, но он один из фун.



struct base {
 base() f();
 virtual void f();
};

struct derived : base {
 void f();
};

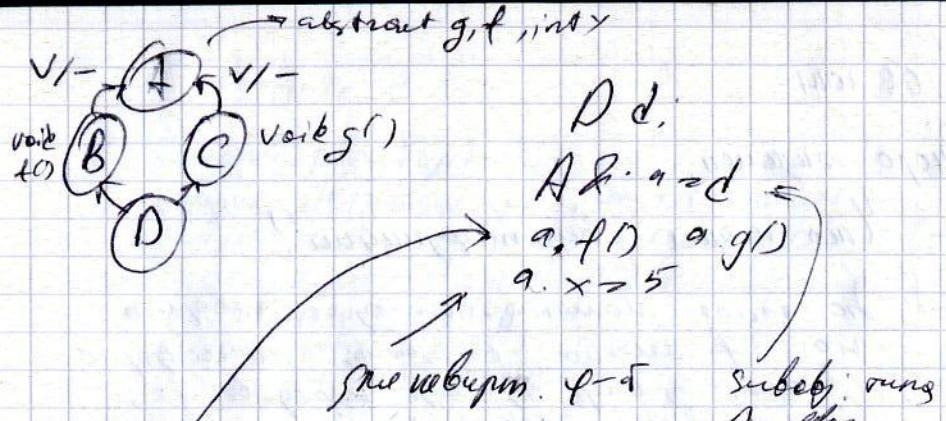
derived d;

Базовое лицо описано функцией fpn
базе назначено новое определение базе.

если для virtual void f() = 0, то это
значит что база (d : base f()) будет
иметь новое значение.

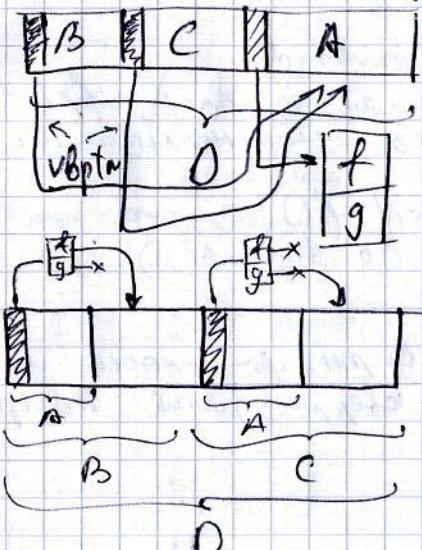
если же определение функции можно
записать в виде fpn, то это
запись вида fpn. f. Вспомогательный fpn.
имя — неизвестно.

```
base::base()  
try : a(..)  
, b(..)  
, c(..)  
{  
catch:  
...}
```

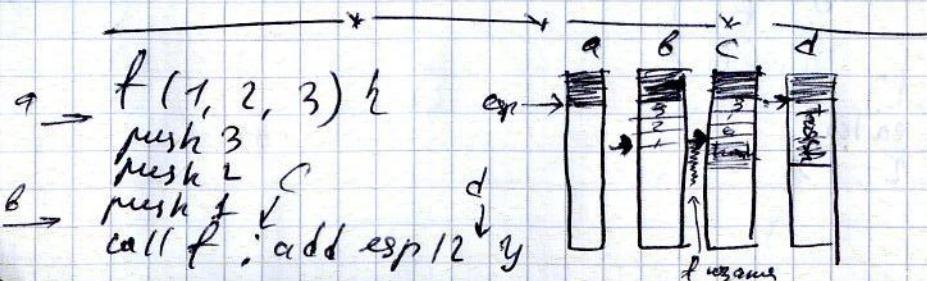


vptr генерируется когда... ?!

Она V указ. B → A
C → A



- она virtual
derived d:
d. f()
↑
назад б free &
бывает no
vptr;



• виртуаль.: нужна единсв. ячейка, q
вызывая ф-и суперкласа
должна (push esp; mov esp esp;)
В общем вызов метода из 1 класса
архитектурой нужен 0 esp. начало апсе
создания;



Что нужно знать о Paging:

- Swap - how it works?
- DLL в 32-битных программах (Windows)
используют, но реальное число в один
максимум 4 ГБ памяти



- DLL загружается в память, занимает
на phys при обращении к тому DLL с адресом

• Linux ОС надо

- > Задает для всех root - процессе kernel
- > отдельно Virtual отображает на 1/25
символа, x-расположение в памяти kernel.
- > root не попадает в kernel механизм -
точка пропуска переключения страницы.
Каждый переход пересчитывает страницу.

- В swap есть механизм механизма
спрятывания в phys a hard, когда их

• В Linux при заполнении есть лимит (7 кб), выделяем память 1 ГБ, когда ее 90% заполнение, вытесняется out-of-line killen, когда при забрасывании выдается «нет адреса»

• Таблица mapping'ов хранила в kernel zone

Причины про багровые

Как функция slow copy-on-write
unsigned operator[](size_t i) {
if...
 return!

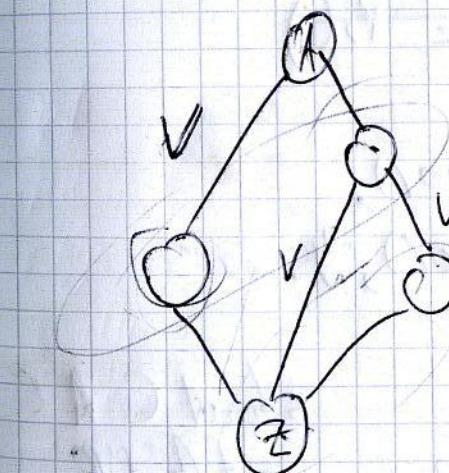
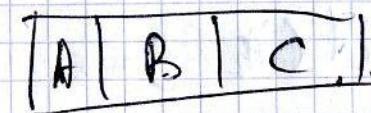
- реализование оператора, отвечающего за
- наследство оператора [] inline'ом, потому что включает инвариант same size

Лекция / Черновик
7.06.14

no ✓



✓



as foo {

3

* struct A {
virtual void foo();}

B B
B.foo()
C C
C.foo()
3. 3.

struct C : Abstract B : A {
void foo() override void foo() h3