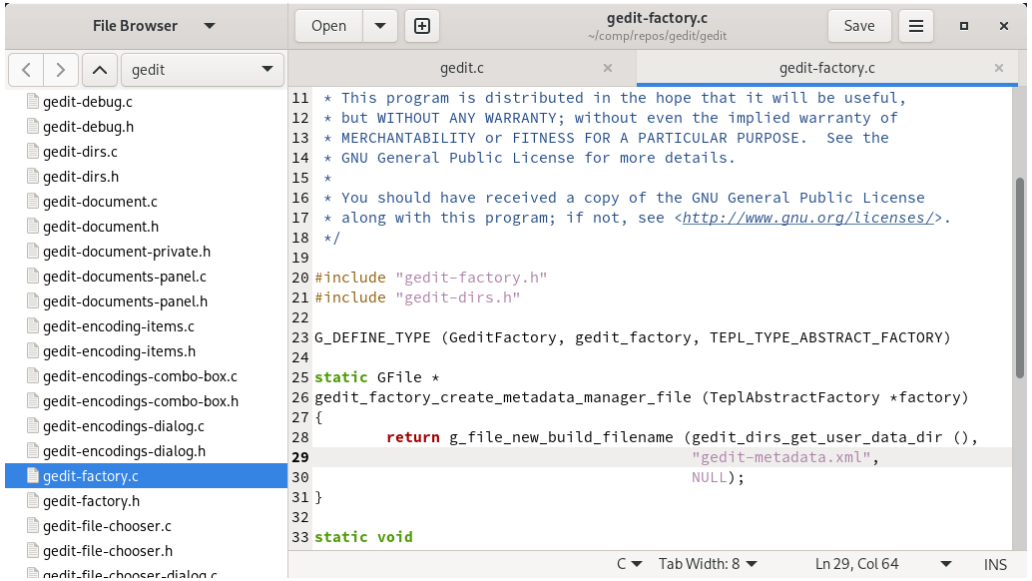
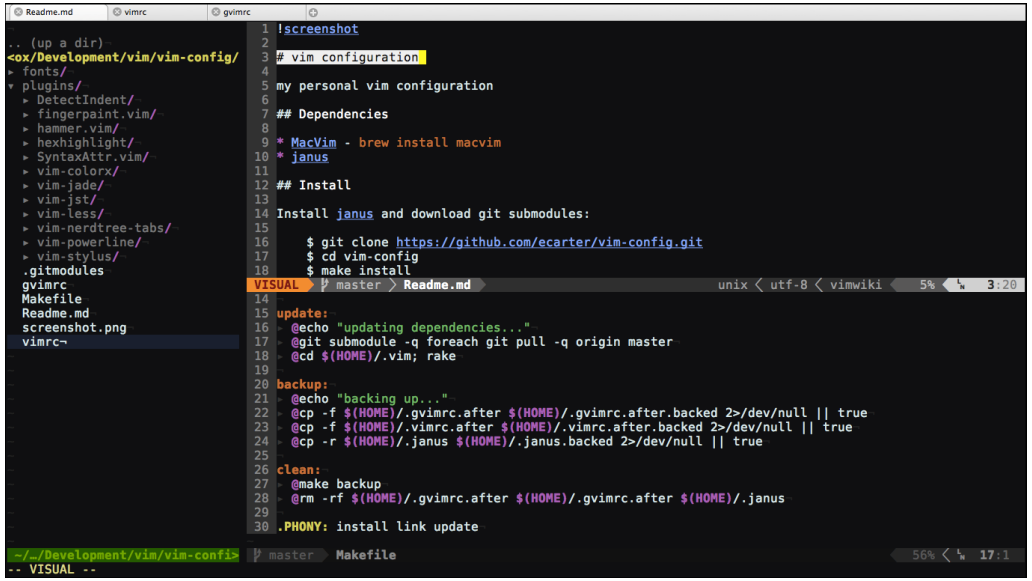
Лекция 2
Данные в языке Си.
Базовые типы данных языка Си.

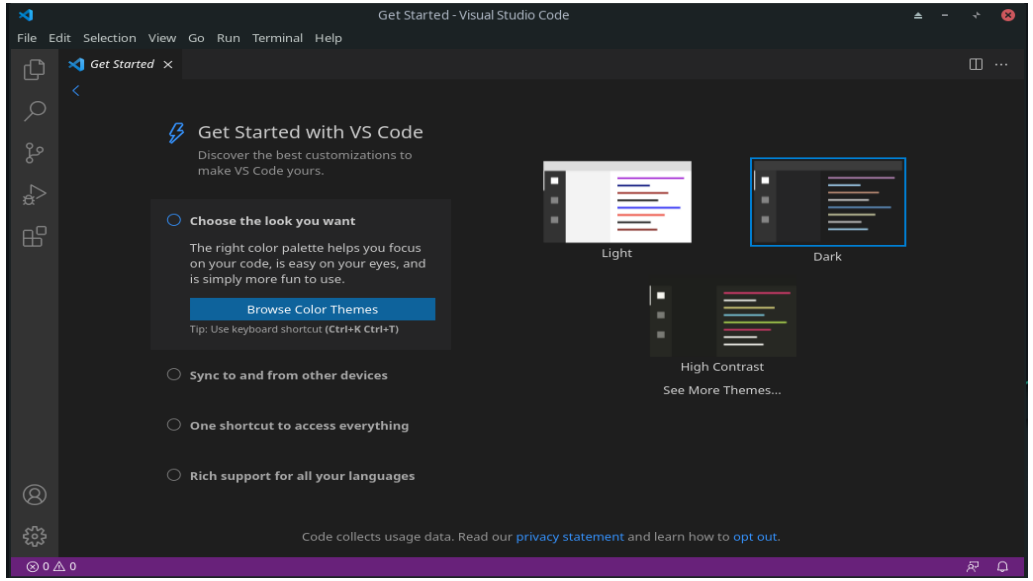
Ревун Артем Леонидович
ст. преп. Кафедры вычислительных систем

Курс «Программирование», осенний семестр
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)

Проблема 1. Текстовый редактор в Linux







Проблема 2. Каркас простейшей программы

```
// C program template (snippet)
// Символы "//" используются для комментариев
/* Или поточный комментарии */

#include <stdio.h> //Библиотеки содержащие /
                   //Функции ввода-вывода
int main()         //Точка входа
{
    // Код программы
    printf("Компилятор - это серьёзное дело");
    return 0;      //Код успешного завершения программы
}
```

- Программа на языке Си представляет собой один или несколько *текстовых файлов* с расширением **".c"**.
- Дополнительно могут использоваться *заголовочные файлы* с расширением **".h"** (2 семестр)
- Для преобразования текстового файла в исполняемый файл используется компилятор **gcc**.

Проблема 3. Компиляция

Мы, всё же пишем на Си

- **Компилятор** – это программа, которая считывает текст программы, написанной на одном языке – исходном, и транслирует (переводит) его в эквивалентный текст на другом языке – целевом
- **Интерпретатор** – программа, которая вместо получения целевой программы, как в случае транслятора, непосредственно выполняет операции, указанные в исходной программе, над входными данными, предоставляемым пользователем
- **Смешанный подход** – компиляция исходного кода в байт-код для его последующей интерпретации

gcc <имя_файла_с_исходным_кодом>.c –o <имя_программы_после_компиляции>

Вызываем gcc

Укажем путь и название файла с исходным кодом

Укажем название исполняемого файла, чтобы знать как к нему обратится

Проблема 3. Компиляция (продолжение)

1) `gcc -E main.c -o main.ii` ←Препроцессинг

Результат→

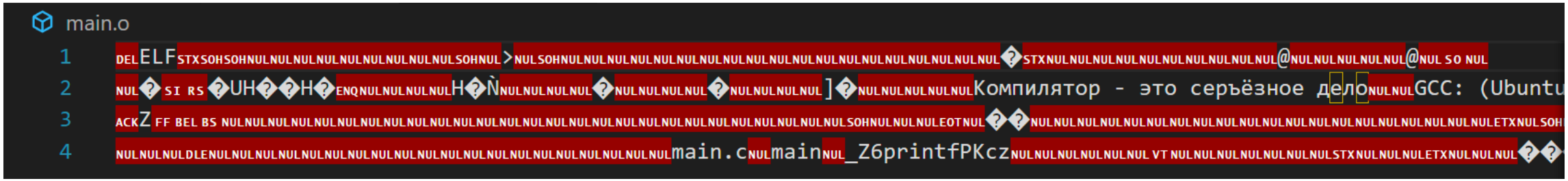
```
743   # 6 "main.c"
744   int main()
745   {
746
747       printf("Компилятор - это серьёзное дело");
748       return 0;
749   }
```

2) `gcc -S main.ii -o main.s` ←Компиляция (машинный код)

```
10    main:
11    .LFB0:
12        .cfi_startproc
13        endbr64
14        pushq   %rbp
15        .cfi_def_cfa_offset 16
16        .cfi_offset 6, -16
17        movq    %rsp, %rbp
18        .cfi_def_cfa_register 6
```

Проблема 3. Компиляция (продолжение)

3) `as main.s -o main.o` ←Ассемблирование



4) `gcc main.o -o main` ←Компоновка (линковка)

```
./main
 Компилятор - это серъёзное дело
```

Программа на языке Си

Процесс написания программы на языке Си сводится к следующим шагам.

1. **Описание данных**: язык Си требует, чтобы каждая область памяти в программе была явно описана. Описание заключается в указании *имени области* и ее *типа данного*. Тип данного определяет какие данные могут храниться в переменной (целые или вещественные) и какой размер ячейки в байтах (следовательно и диапазон значений).

2. **Преобразование данных**: преобразование данных осуществляется с помощью базовых конструкций языка: *следование*, *ветвление*, *цикл*.

3. **Организация взаимодействия с окружающей средой**:

1) с локальной операционной системой для *настройки параметров исполнения* и *ввода-вывода*;

2) другими процессами на локальной или удаленной машине для осуществления *ввода-вывода*.

Алгоритм

Алгоритм (**algorithm**) – это конечная последовательность инструкций исполнителю, в результате выполнения которых обеспечивается получение из входных данных требуемого выходного результата (решение задачи)

Формы записи алгоритма:

- словесная или вербальная (языковая, формульно-словесная);
- псевдокод (формальные алгоритмические языки);
- графическая (блок-схемы).

Запись алгоритма на формальном языке исполнителя называется **программой** (program)

Свойства алгоритма

- **Дискретность** – алгоритм представляется как последовательность инструкций исполнителя. Каждая инструкция выполняется только после того, как закончилось выполнение предыдущего шага
- **Конечность** (результативность) – алгоритм должен заканчиваться после выполнения конечного числа инструкций
- **Массовость** – алгоритм решения задачи должен быть применим для некоторого класса задач, различающихся лишь значениями входных данных
- **Детерминированность** (определенность) – каждый шаг алгоритма должен быть точно определен – записан на формальном языке исполнителя. Детерминированность обеспечивает одинаковость результата, получаемого при многократном выполнении алгоритма на одном и том же наборе входных данных

Парадигмы программирования

Парадигма программирования - это способ программирования, не зависящий от конкретного языка. Парадигма определяет, какие структуры использовать и когда их использовать.

- Императивное программирование
- Декларативное программирование

- Структурное программирование
- Функциональное программирование
- Объектно-ориентированное программирование
- Логическое программирование

Язык Си

Реализует **императивную парадигму** программирования: программа представляет собой набор директив предназначенных для исполнителя.

Ориентирован на **структурное программирование**. Данная парадигма предполагает следующее:

- программа состоит из трех базовых конструкций: *следование*, *ветвление*, *цикл*;
- базовые конструкции могут быть вложены друг в друга;
- логически законченные фрагменты программы оформляются в виде подпрограмм.

Архитектура ЭВМ Дж. фон Неймана

Основные черты архитектуры ЭВМ Дж. Фон Неймана

- **Принцип однородности памяти** – команды и данные хранятся в одной и той же памяти (внешне неразличимы)
- **Принцип адресности** – память состоит из пронумерованных ячеек, процессору доступна любая ячейка

The EDVAC as installed in Building 328 at the Ballistics Research Laboratory, Maryland USA // Wikipedia

Характеристики переменных

Переменная – область памяти, предназначенная для хранения данных.

В языках высокого уровня переменная характеризуется:

- **именем** (идентификатором), которое позволяет компилятору отличать данную ячейку от множества других, используемых в программе;
- **типом данных**, который определяет какую информацию можно хранить в данной ячейке (целые или вещественные числа). Тип данного определяет:

1) формат представления информации;
2) диапазон хранимых значений (размер ячейки памяти);
3) набором допустимых операций.

Программа на языке Си

Области памяти, доступные в языке Си, можно классифицировать следующим образом.

1. **Переменные базовых типов данных.**
2. *Массивы базовых типов данных.(конец первого семестра)*
3. *Переменные и массивы сложных типов данных.*
4. *Неименованные области (динамическая память).*

На первом этапе освоения языка будем использовать только переменные базовых типов.

Описание такой переменной имеет следующий формат:

<*тип*> <*имя*1> [=<*знач*>] {, <*имя*2> [=<*знач*>] };

Например:

```
int i, test, flag;
float var, discriminant;
```

Идентификаторы

Компьютерная программа средней сложности содержит тысячи программных объектов. Для того, чтобы уникально идентифицировать их используются **"идентификаторы".**

Идентификатор представляет собой последовательность строчных ("**a – z**") и прописных ("**A – Z**") букв латинского алфавита, а также цифр ("**0 – 9**") и знака подчеркивания ("_").
Идентификатор **начинается** либо **с буквы**, либо **со знака нижнего подчеркивания**.

Идентификаторы используются для:

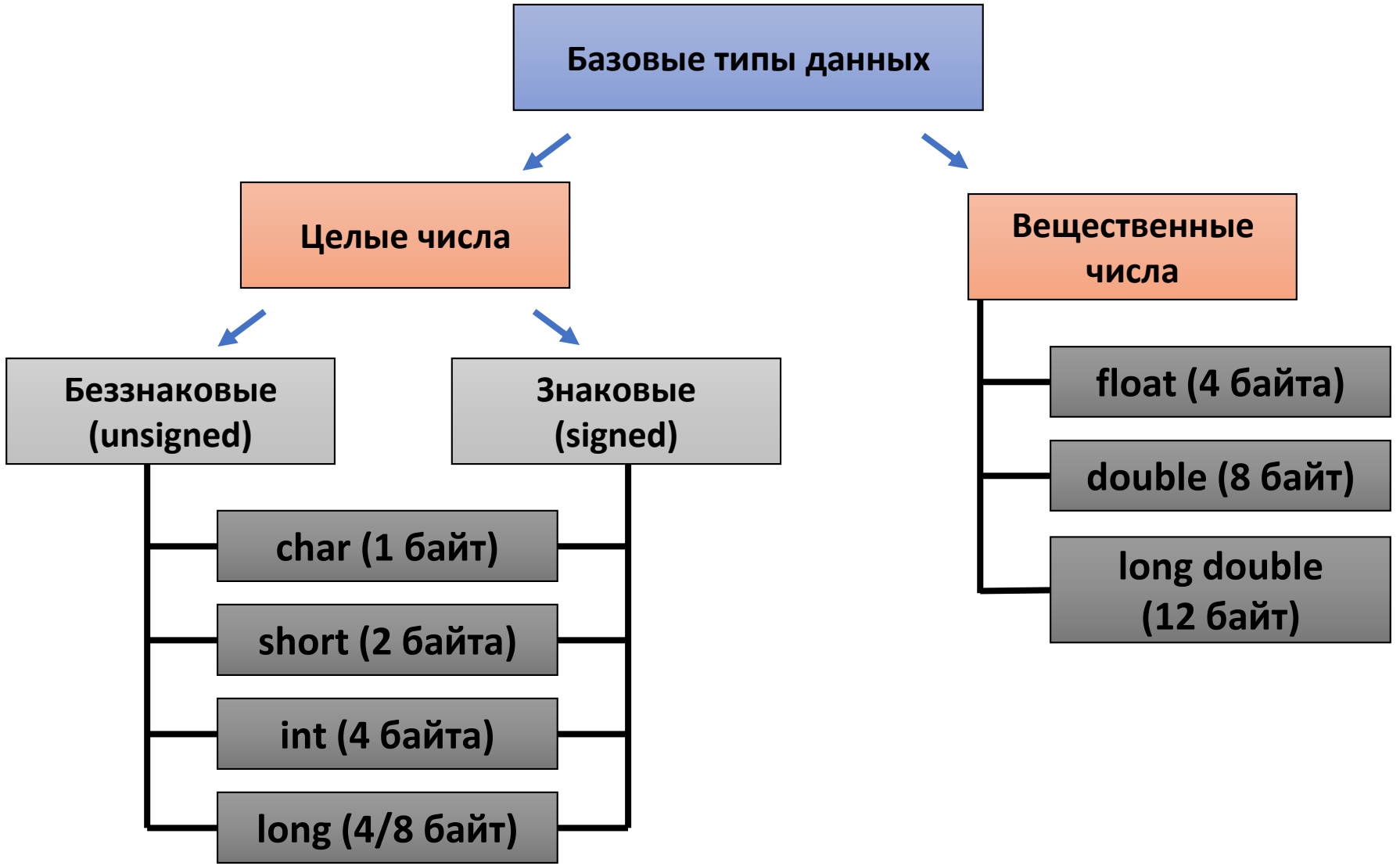
1) формирования имен объектов языка: *переменных* (ячеек памяти для хранения данных) *и функций* (групп инструкций программы).
2) ключевых слов языка: *типы данных* (**int**, **char**, **float**), *управляющие конструкции* (**for**, **while**, **if**).

Примеры идентификаторов

Идентификатор – последовательность строчных ("**a – z**") и прописных ("**A – Z**") букв латинского алфавита, а также цифр ("**0 – 9**") и знака подчеркивания ("_").
Идентификатор **начинается** либо **с буквы**, либо **со знака нижнего подчеркивания**.

Допустимые	Недопустимые
i	123
int	_@
test1234	test~
Q1W2E3R4	1Q1W2E3R4
S123456789	-S123456789
_123456789	_1234=56789
John_Deer	John Deer

Базовые типы данных (скалярные)



Представление целых чисел в Си

Беззнаковые целые

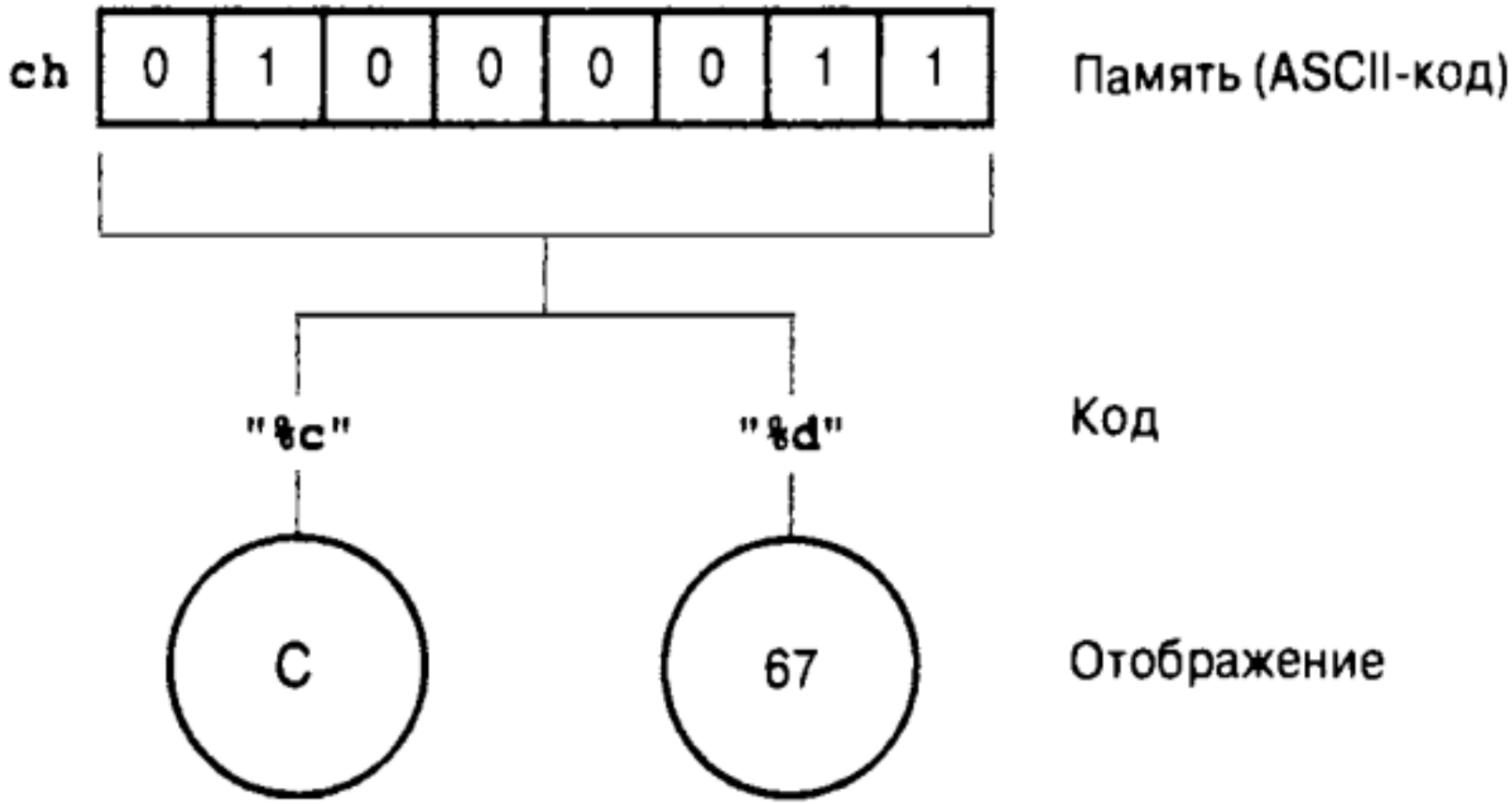
Обозначение	Размер, байт	Диапазон значений
unsigned char	1	[0; 255]
unsigned short	2	[0; 65535]
unsigned int	4	[0; 4294967295]
unsigned long	4 или 8	[0; $2^{64} - 1$] (8 байт)

Представление целых чисел в Си (продолжение)

Знаковые целые

Обозначение	Размер, байт	Диапазон значений
[signed] char	1	[-128; 127]
[signed] short	2	[-32768; 32767]
[signed] int	4	[-2147483648; 2147483647]
[signed] long	4 или 8	$[-2^{63}; 2^{63} - 1]$ (8 байт)

Представление целых чисел в Си (char)



Представление целых чисел в Си (char)

TABELA ASCII

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Представление вещественных чисел в Си

Идентификатор	Размер, байт	Диапазон значений
float	4	от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)
double	8	от $\pm 1.7 \cdot 10^{-308}$ до $\pm 1.7 \cdot 10^{308}$ (~ 15 значащих цифр)
long double	12	от $\pm 1.2 \cdot 10^{-4932}$ до $\pm 1.2 \cdot 10^{4932}$ (~22 значащие цифры)

Создание и инициализация переменных

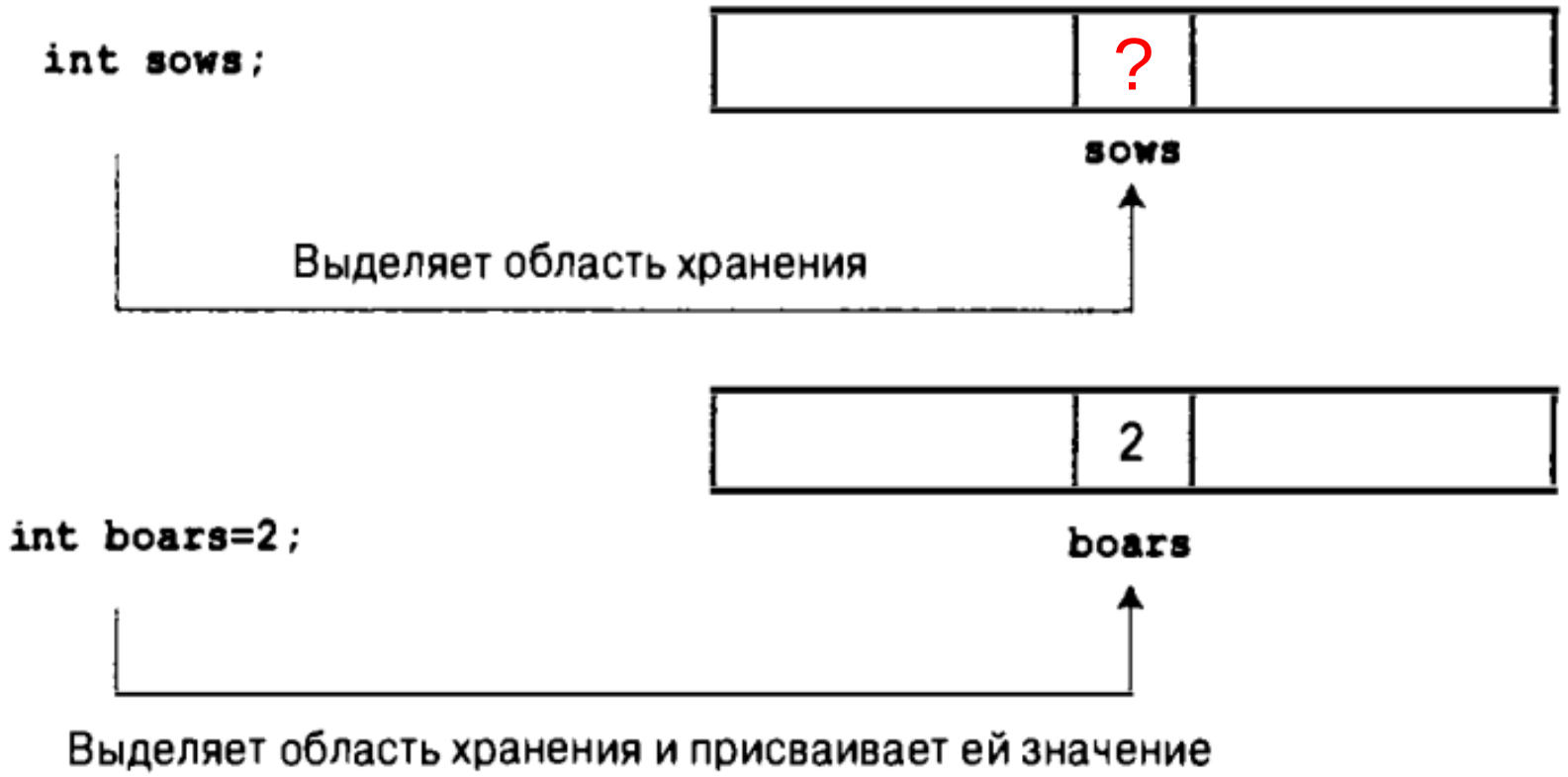


Рис. 3.4. Определение и инициализация переменной

Представление числа в памяти

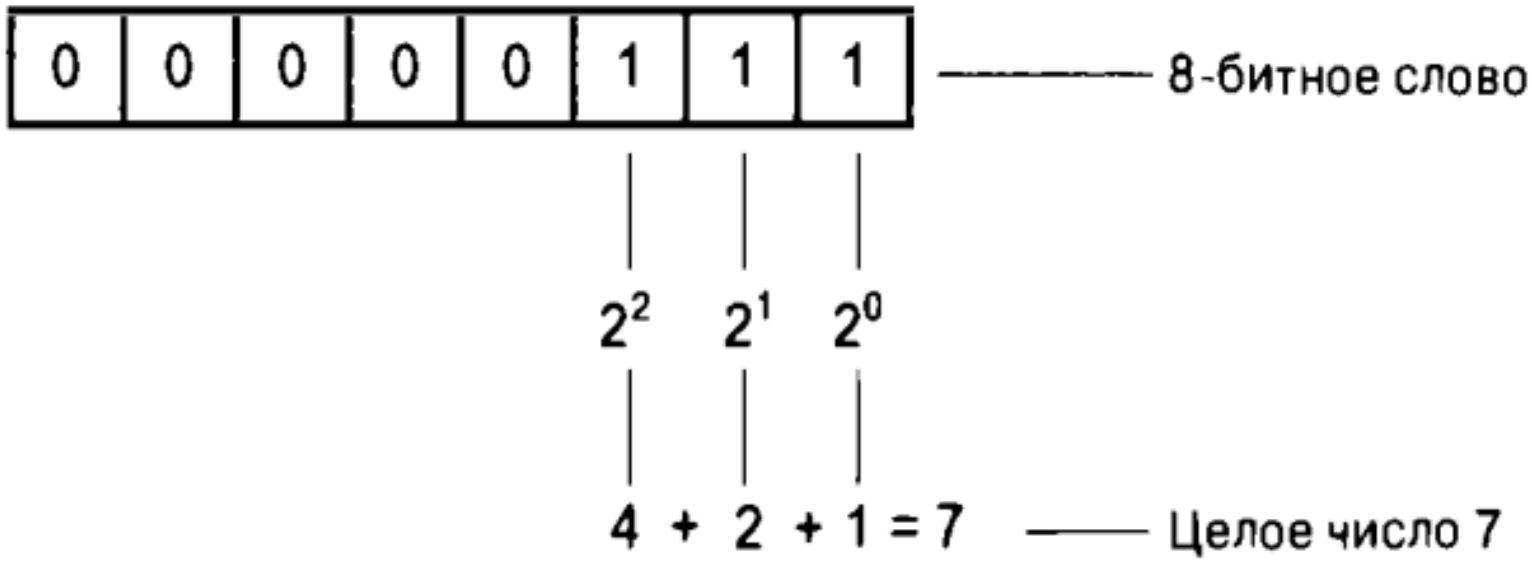


Рис. 3.2. Хранение целого числа 7 в двоичном коде

Переполнение ячейки памяти

Знаковые целые

```
int i = 2147483647;
printf("%d %d %d\n", i, i+1, i+2);
2147483647 -2147483648 -2147483647
```

на примере (signed) int 32bit

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Переполнение ячейки памяти

Беззнаковые целые

```
unsigned int j = 4294967295;
printf("%u %u %u\n", j, j+1, j+2);
```

на примере (unsigned) int 32bit

```
4294967295 0 1
```

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Компиляция типов данных

int.c

```
#include <stdio.h>

int main()
{
    int a = 5, b = 6, c;
    c = a + b;
}
```

float.c

```
#include <stdio.h>

int main()
{
    float a = 5, b = 6, c;
    c = a + b;
}
```

gcc -S int.c

int.s

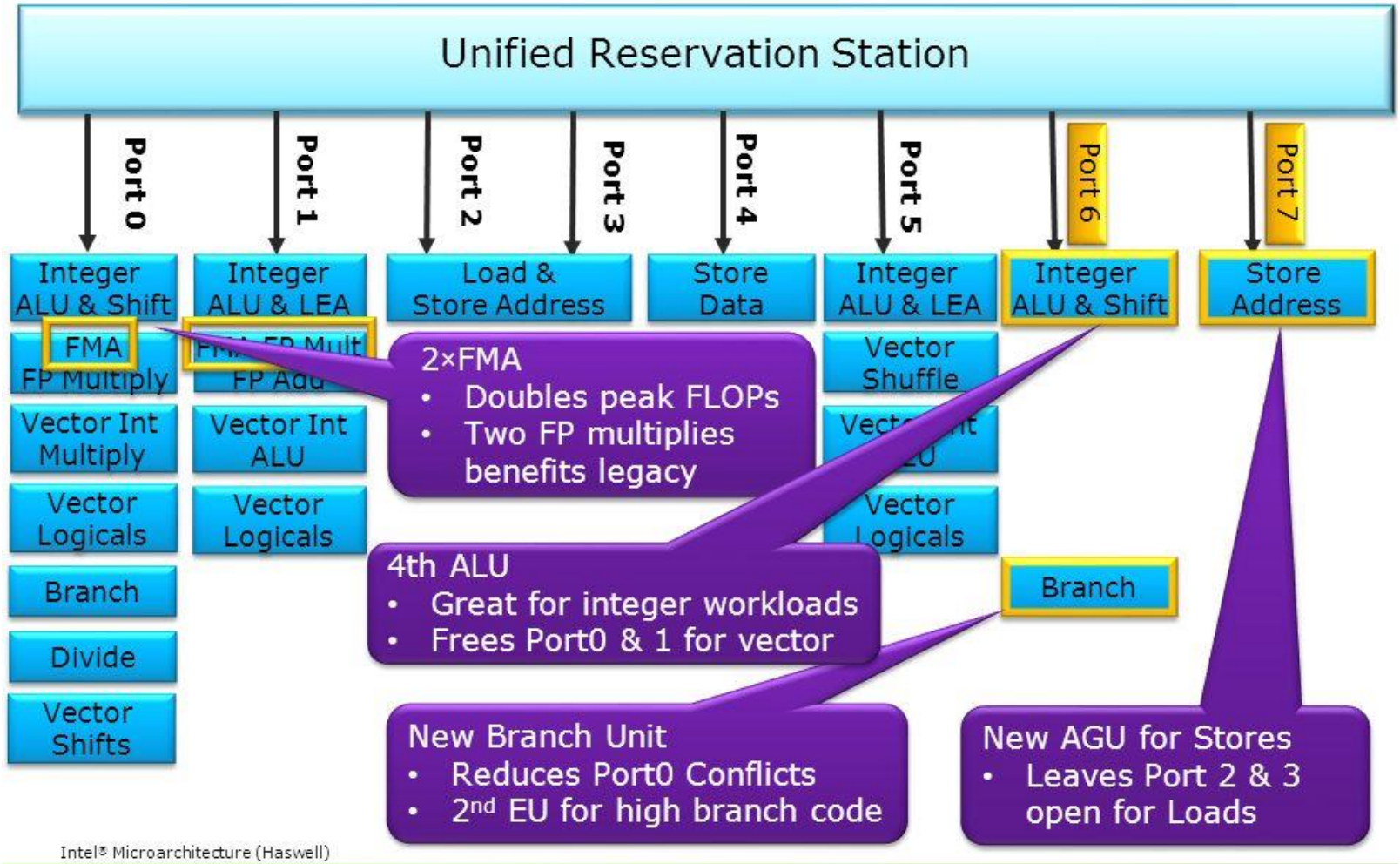
```
    ...
    movl   -8(%rbp), %eax
    movl   -12(%rbp), %edx
    addl   %edx, %eax
    ...
```

gcc -S -mno-sse float.c

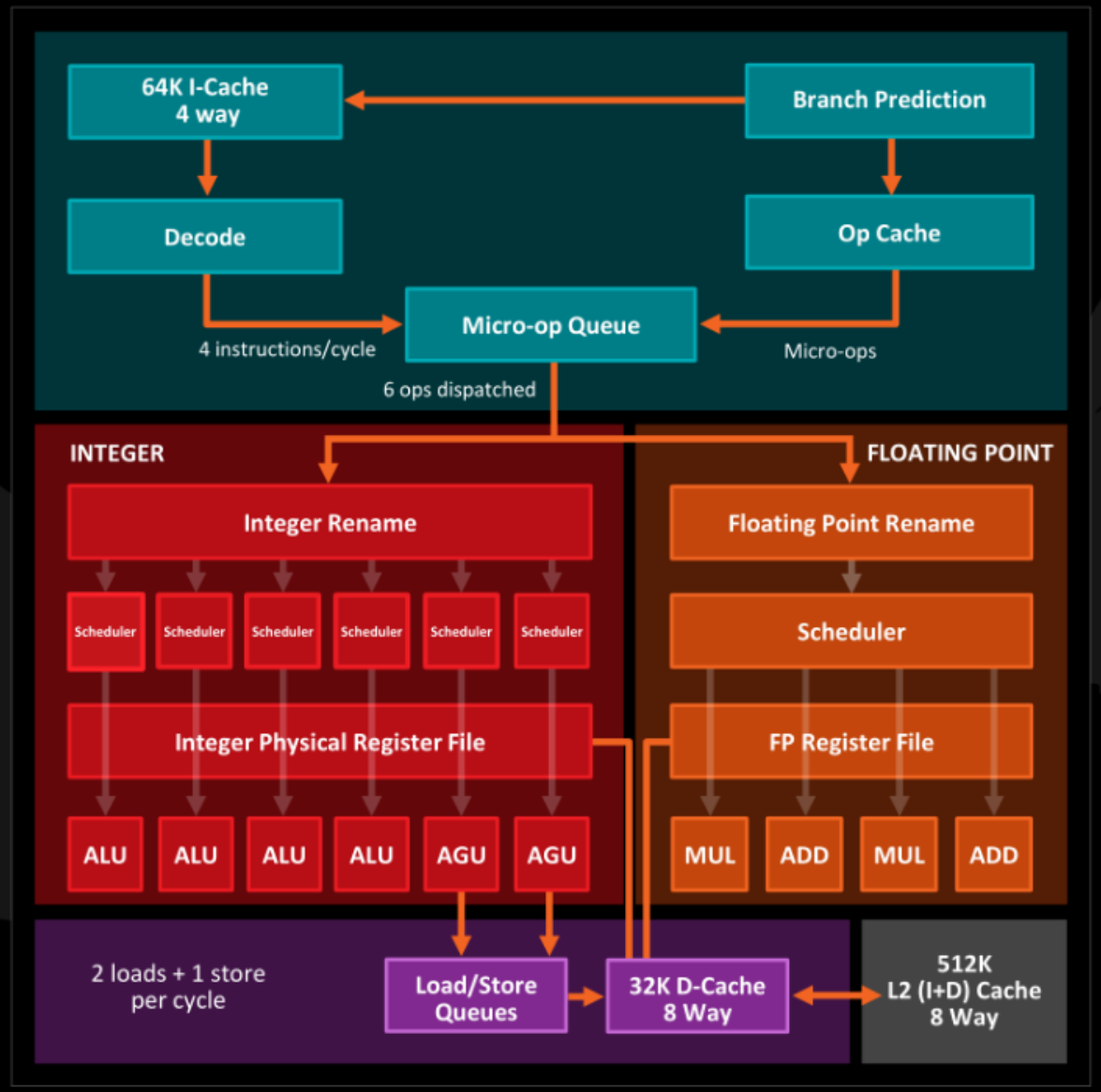
float.s

```
    ...
    flds     -12(%rbp)
    fadds    -8(%rbp)
    fstps    -4(%rbp)
    ...
```

Haswell Execution Units



Intel® Microarchitecture (Haswell)



ZEN MICROARCHITECTURE

- Fetch Four x86 instructions
- Op Cache instructions
- 4 Integer units
  - Large rename space – 168 Registers
  - 192 instructions in flight/8 wide retire
- 2 Load/Store units
  - 72 Out-of-Order Loads supported
- 2 Floating Point units x 128 FMACs
  - built as 4 pipes, 2 Fadd, 2 Fmul
- I-Cache 64K, 4-way
- D-Cache 32K, 8-way
- L2 Cache 512K, 8-way
- Large shared L3 cache
- 2 threads per core

Некоторые команды процессора (архитектура x86)

ЦЕЛОЧИСЛЕННЫЕ

Арифметические

ADD – сложение
SUB – вычитание
MUL – умножение
DIV – деление нацело
IMUL – *знаковое* умножение
IDIV – *знаковое* деление

Логические

AND – логическое И
OR – логическое ИЛИ
XOR – исключающее ИЛИ

Сдвиги

SHL/SHR – сдвиг влево/вправо
SAL/SAR – арифметический (*знаковый*) сдвиг влево/вправо

Над числами с ПЛАВАЮЩЕЙ ТОЧКОЙ

Арифметические

FADD – сложение
FSUB – вычитание
FMUL – умножение
FDIV – деление
FIMUL – умножить вещественное число на целое
FIDIV – разделить вещественное число на целое
FSIN – вычислить синус
FSINCOS – вычислить синус и косинус

Операции вывода printf()

```
#include <stdio.h>
int main(void)
{
    int x = 100;
    printf("десятичное = %d; восьмеричное = %o; шестнадцатеричное = %x\n", x, x, x);
    printf("десятичное = %d; восьмеричное = %#o; шестнадцатеричное = %#x\n", x, x, x);
    return 0;
}
```

```
десятичное = 100; восьмеричное = 144; шестнадцатеричное = 64
десятичное = 100; восьмеричное = 0144; шестнадцатеричное = 0x64
```

Операции вывода printf()

Последовательность	Описание
\a	Предупреждение (стандарт ANSI C)
\b	Возврат на одну позицию влево
\f	Перевод страницы
\n	Новая строка
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\\	Обратная косая черта (\)
\'	Одиночная кавычка (')
\"	Двойная кавычка (")
\?	Знак вопроса (?)
\0*oo*	Восьмеричное значение (*o* представляет восьмеричную цифру)
\x*hh*	Шестнадцатеричное значение (*h* представляет шестнадцатеричную цифру)

Примеры целочисленных литералов

Примеры целочисленных констант			
Тип	**Шестнадцатеричная**	**Восьмеричная**	**Десятичная**
char	\0x41	\0101	–
int	0x41	0101	65
unsigned int	0x41u	0101u	65u
long	0x41L	0101L	65L
unsigned long	0x41UL	0101UL	65UL
long long	0x41LL	0101LL	65LL
unsigned long long	0x41ULL	0101ULL	65ULL

```
printf("%d\t%#o\t%#x \n", 0x41, 0101, 65);
printf("%d\t%d\t%d \n", 0x41, 0101, 65);
```

```
65      0101    0x41
65      65      65
```

Примеры вещественных литералов

Число	Научная форма записи	Экспоненциальная форма записи
1 000 000 000	1.0×10^{9}	1.0e9
123 000	1.23×10^{5}	1.23e5
322,56	3.2256×10^{2}	3.2256e2
0,000056	5.6×10^{-5}	5.6e-5

```
    printf("%f\t%f\t%Lf \n", 322.56, 32256e-2, 32256e-2L);
    printf("%f\t%e \n", 3.1415, 3.1415, 3.1415, 3.1415);
```

```
322.560000      322.560000      322.560000
3.141500        3.141500e+00
```

Задания на практику | Home work

- Познакомиться с командой scanf() (stdio.h)
- Познакомиться с целочисленным типом bool
- Узнать о потери значимости в операциях с плавающей запятой
- Узнать об ошибках округления данных с плавящей запятой

- Смотреть вопросы для самоконтроля!!!(114 стр, page 94)
- Упражнения по программированию
  RU – страница-116
  EN – page-97
- А также https://eios.sibsutis.ru/pluginfile.php/316656/mod_resource/content/1/03-type-support.pdf

Спасибо за внимание

Ревун Артем Леонидович
ст. преп. Кафедры вычислительных систем

Курс «Программирование», осенний семестр, 2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)