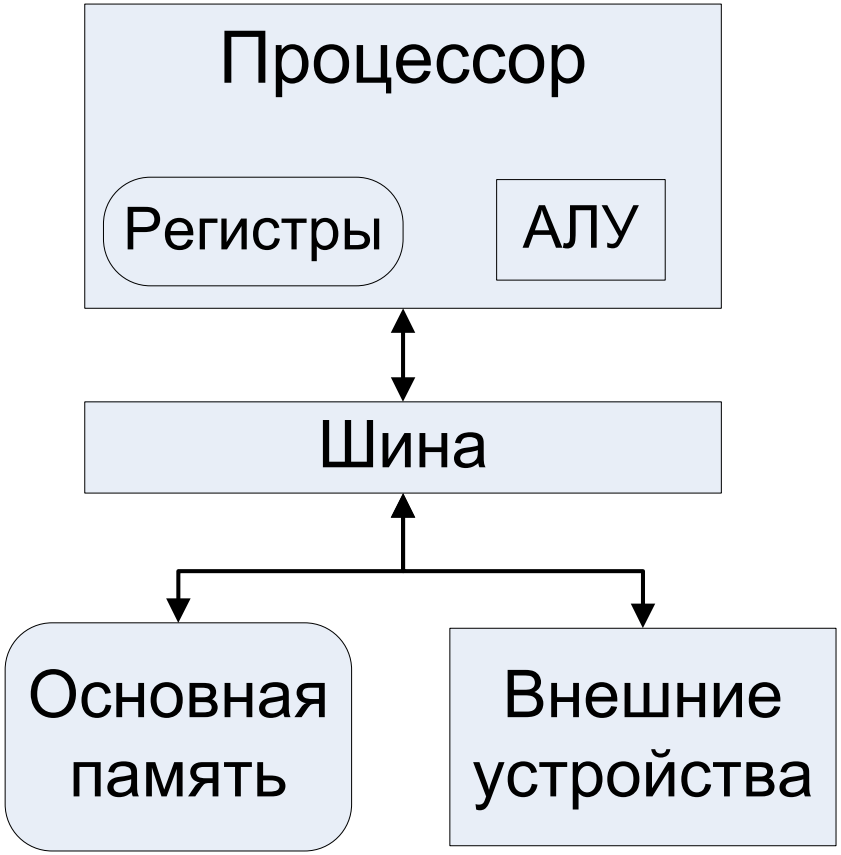
Курс «Алгоритмы и алгоритмические языки»

Лекции 4-5

Критика модели вычислений Тьюринга

- Медленная (неускоряемая)
  - *частые копирования данных*: у нормальных МТ каждое неэлементарное действие выполняется над крайними правыми словами ленты
  - отказ от нормальных вычислений приведет к постоянному поиску данных и усложнит алгоритм
  - *буквальная обработка*: длина описания программы пропорциональна числу букв сообщения, просматриваемых при выполнении алгоритма

Схема простейшего компьютера



Язык программирования Си

- Си разрабатывался как язык для реализации первой в мире универсальной операционной системы UNIX
- 1973 – первая версия Си
- 1978 – выход книги Б. Кернигана и Д. Ритчи «Язык программирования Си» (K&R C). Русский перевод вышел в 1985 году.
- 1989 – первый стандарт ANSI C (C89)
- 1999 – стандарт C99
- 2011 – стандарт C11 (ранее назывался C1X)

Введение в язык программирования Си

Характеристики языка Си

- Императивный язык
- Удобный синтаксис
- Позволяет естественно оперировать «машинными» понятиями
- Переносимость на уровне исходного кода
  - Конфигурируемость
- Хорошие системные библиотеки
- Хорошие оптимизирующие компиляторы

Первая программа на Си

```
#include <stdio.h>

int main (void)
{
    printf ("Hello, world\n");
    return 0;
}
```

Программа:

- объявления переменных или функций
- определения функций

Первая программа на Си

```
#include <stdio.h>

int main (void)
{
    printf ("Hello, world\n");
    return 0;
}
```

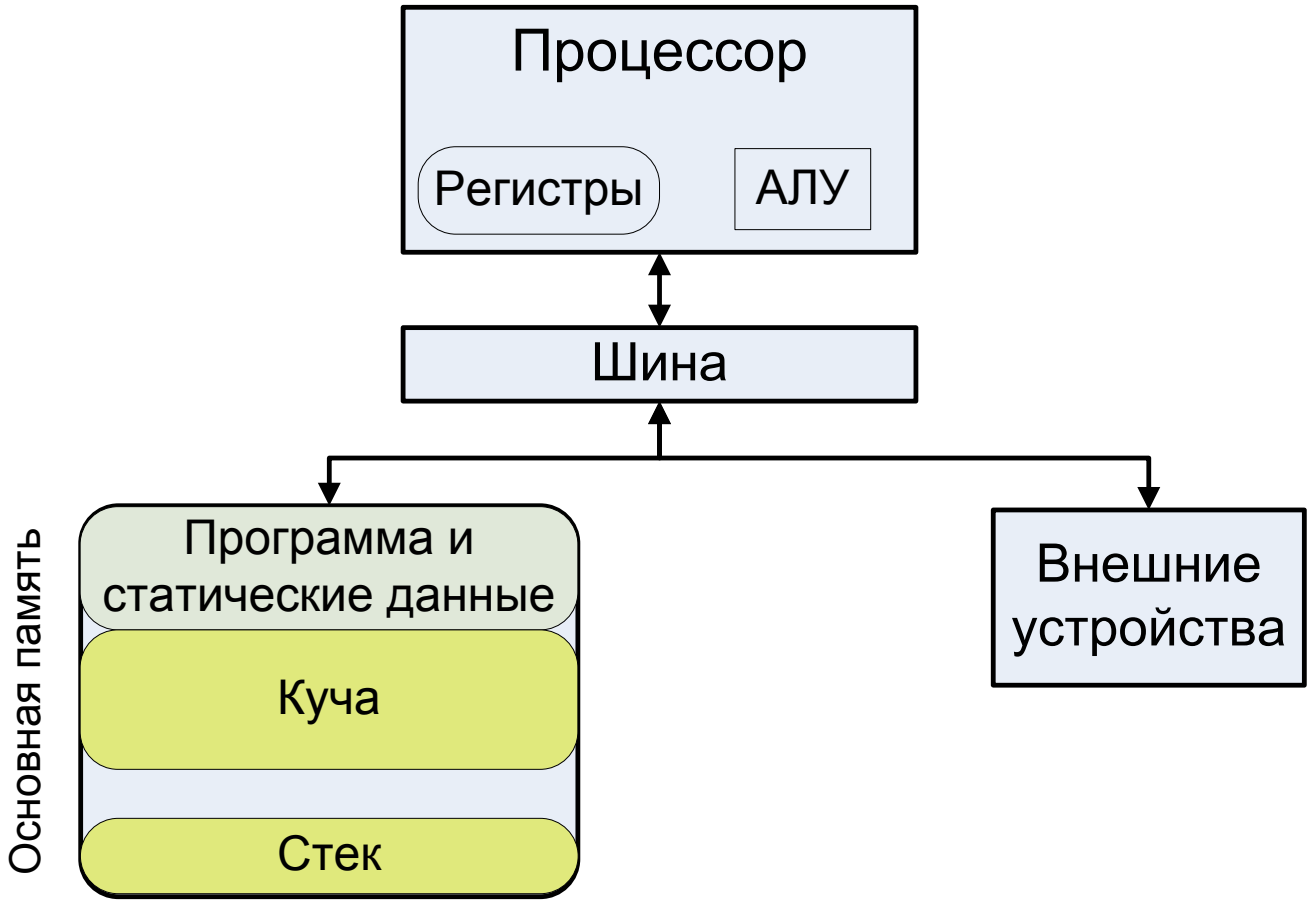
Директивы препроцессора

Системные библиотеки

Строковые константы

Управляющие последовательности

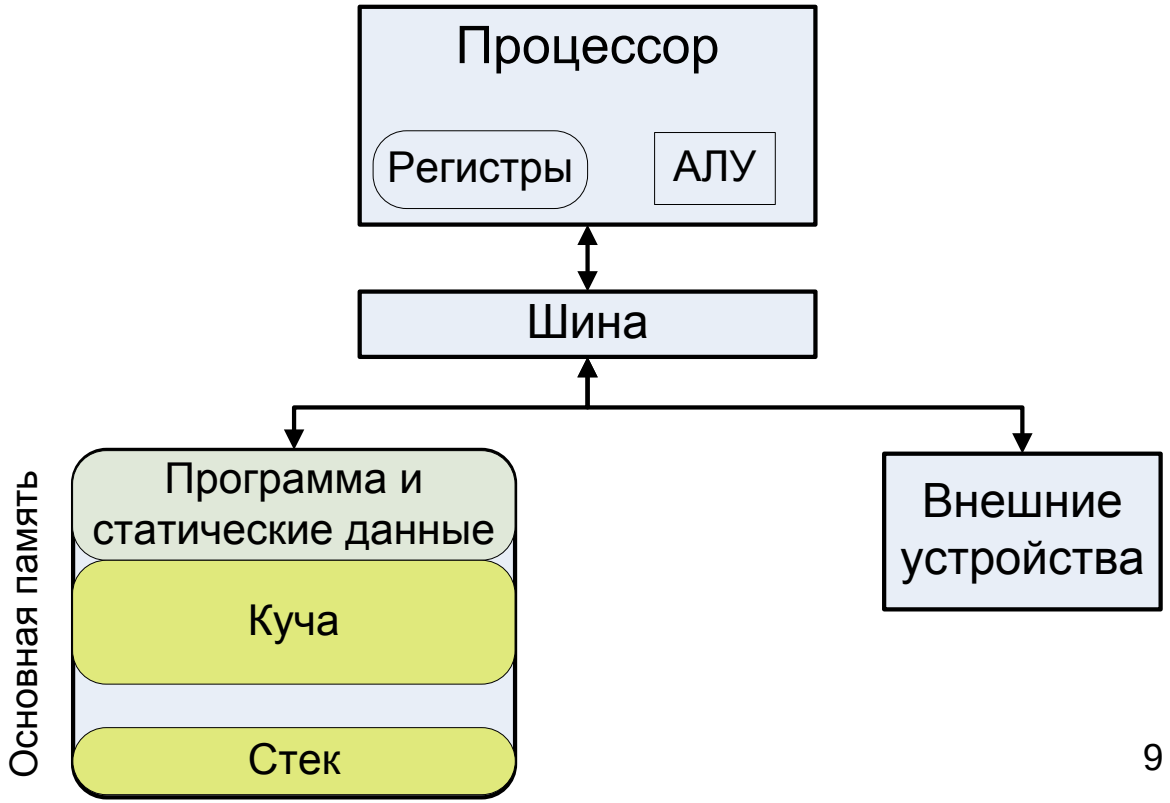
Си-машина


Процессор
Регистры
АЛУ
Шина
Основная память
Программа и статические данные
Куча
Стек
Внешние устройства

Си-машина

Классы памяти

- Регистровые переменные
- Автоматические переменные
- Статические переменные
- Глобальные переменные



Типы данных

- Базовые типы данных: **`char`** (*символьный*), **`int`** (*целый*), **`float`** (*с плавающей точкой*), **`double`** (*двойной точности*), **`void`** (*без значения*)
- Модификаторы базовых типов: **`signed`**, **`unsigned`**, **`long`**, **`short`**, **`long long`** (C99)
  - к типу **`int`** применимы все модификаторы
  - к типу **`char`** – только **`signed`** и **`unsigned`**
  - к типу **`double`** – только **`long`** (C99)
- Представление целых чисел: позиционная двоичная система
  - Байты в представлении числа идут подряд
  - Порядок байт не гарантируется (big/little endian)
  - Отрицательные числа часто представляются в дополнительном коде

Типы данных

- **`sizeof`** – размер типа (любого объекта типа)
  - **`int x -> sizeof(x) == sizeof(int)`**
  - Файл **`limits.h`** задает минимальные и максимальные значения целых типов
  - **`sizeof(char)`**$\geq$ **`1`**
    $\wedge$
    `sizeof(short)`$\geq$ **`2`**
    $\wedge$
    `sizeof(int)` $\geq$ **`2`**
    $\wedge$
    `sizeof(long)` $\geq$ **`4`**
    $\wedge$
    `sizeof(long long)`$\geq$ **`8`**

Типы данных

- Тип **`_Bool`** (C99, значения 0/1, целый беззнаковый)
  - Необходимо включить **`stdbool.h`** для объявлений **`bool`**, **`true`**, **`false`**
- Тип **`_Complex`** (C99, **`float/double/long double`**)
  - Необходимо включить **`complex.h`** для объявлений **`complex`**, **`I`** и т.п.
  - Тип **`_Imaginary`** (C99) является необязательным

Переменные

- Переменная = тип + имя + ***значение***

  Каждая переменная является *объектом* программы
- **`Ключевые слова`** (C89 – 32, C99 – C89 + 5) не могут быть именами переменных
- Объявление переменной:

  тип список_переменных

  Можно задать класс памяти и начальное значение переменной

Область действия переменной

- Переменная может быть объявлена:
  (1) внутри функции или блока (локальная);
  (2) в объявлении функции (параметр функции);
  (3) вне всех функций (глобальная).
- Область действия (видимости)
  - локальной переменной – блок, в котором она объявлена
  - глобальной переменной – программный файл, начиная со строки объявления
- В одной области действия нельзя объявлять более одной переменной с одним и тем же именем

Область действия переменной и классы памяти

```
#include <stdio.h>
int count;                      /* глобальная переменная */
static int mult = 0;            /* статическая переменная */
int sum (int x, int y)
{
  count++;
  return (x + y) * (++mult);
}
void func (void)
{
  int count;                    /* автоматическая переменная */
  count = count - 2;
}
int main (void)
{
  register int s = 0;           /* регистровая переменная */
  count = 0;
  s += sum (5, 7);
  s += sum (9, 4);
  func ();
  printf ("Сумма равна %d, вызвали функцию %d раз\n", s, count);
  return 0;
}
```

Инициализация переменной

- При объявлении переменной: **`int x = 42;`**
  - автоматические переменные инициализируются каждый раз при входе в соответствующий блок;

    если нет инициализации, они имеют произвольное значение
  - глобальные и статические инициализируются только

    один раз в начале работы программы;

    если нет инициализации, они обнуляются компилятором
  - внешние переменные инициализируются только в том файле, в котором они определяются
  - при инициализации переменной типа с квалификатором **`const`** она является константой и не может изменять свое значение

Литералы

- Литералы задают константы
  - символьные константы **`'c'`**, **`L'%'`**, **`'\0x4f'`**, **`'\020'`**
  - целые константы **`100`**, **`-34l`**, **`1000U`**, **`999llu`**
  - константы с плавающей точкой **`11.123F`**, **`4.56e-4f`**, **`1.0`**, **`11.123`**, **`3.1415926l`**, **`6.626068e-34L`**
  - шестнадцатеричные константы **`0x80`** (*128*)
  - восьмеричные константы **`012`** (*10*)
  - строковые константы **`"a"`**, **`"Hello, World!"`**, **`L"Unicode string"`**
  - специальные символьные константы **`\n`**, **`\t`**, **`\b`**

Операции над целочисленными данными

- ***Арифметические***
  - *Одноместные*:
    изменение знака, или «одноместный минус» (**–**),
    одноместный плюс (**+**).
  - *Двухместные*:
    сложение (**+**), вычитание (**–**), умножение (*****),
    деление нацело (**/**), остаток от деления нацело (**%**).
- ***Отношения*** (результат 0/1 типа **`int`**)
  - больше (**>**), больше или равно (**>=**),
    меньше (**<**), меньше или равно (**<=**)
- ***Сравнения*** (результат 0/1 типа **`int`**)
  - равно **(==),** не равно **(!=)**
- ***Логические***
  - отрицание (**!**), конъюнкция **(&&),** дизъюнкция **(||)**
  - ложное значение – 0, истинное – любое ненулевое
  - “ленивое” вычисление **&&** и **||**

Операции присваивания

- ***Побочные эффекты:*** изменение объекта, вызов функции
- **`lvalue = rvalue`**
  - **`lvalue`** – выражение, указывающее на объект памяти
  - **`rvalue`** – выражение
  - Пример **`a = b = c = d = 0;`**
- ***Укороченное присваивание:*** **`lvalue op= rvalue`**
  где **`op -`** двухместная операция

  Пример **`a += 15;`**
- ***Инкремент и декремент*** (**`++`** и **`--`**)
  - префиксные и постфиксные
- ***Последовательное вычисление:*** операция запятая (**`,`**)

  Пример **`a = (b = 5, b + 2);`**

Точки следования

- ***Побочные эффекты:*** изменение объекта, вызов функции
- ***Точка следования* (*sequence point*)*:*** момент во время выполнения программы, в котором все побочные эффекты предыдущих вычислений закончены, а новых – не начаты
  - первый операнд **&&**, **||**, **,**
  - окончание ***полного*** выражения
  - между двумя точками следования изменение значения переменной возможно не более одного раза

    ```
    (a=2) + (a=3)
    ```

    ```
    i++ + ++i
    ```
  - старое значение переменной читается только для определения нового

    ```
    a = b++ + b
    ```