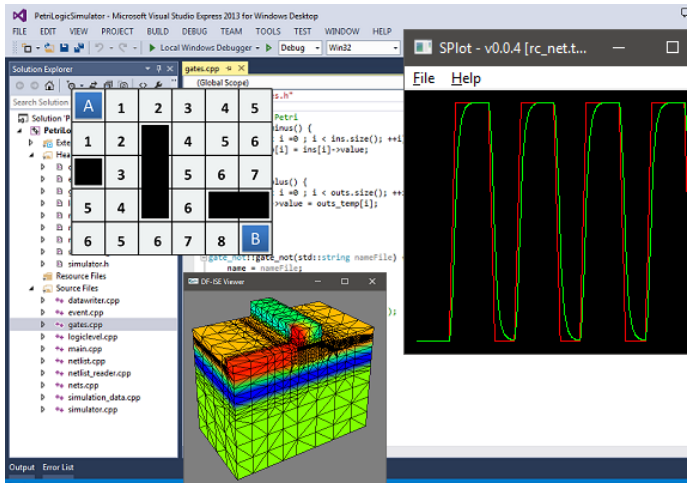
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники (группы ЭН-34-35, каф. ПКИМС)

Программные средства САПР



Лекция 2

Этапы сборки программного обеспечения

Процесс сборки программ

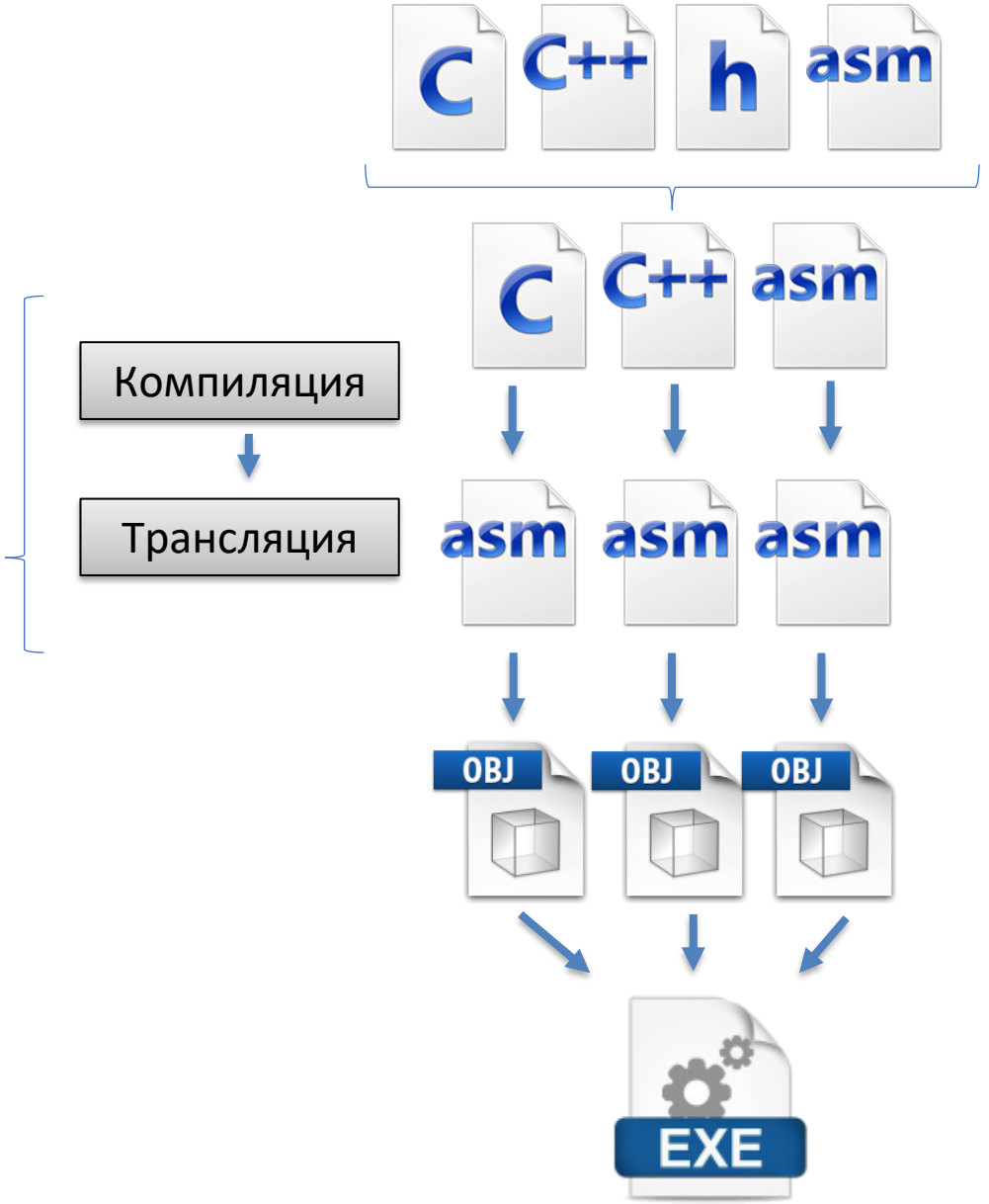
Работа компилятора

Предварительная обработка исходного кода препроцессором

Компиляция исходного кода каждого отдельного файла

Работа линковщика

Линковка нескольких объектных модулей в один файл



Директивы препроцессора (1)

Подключение заголовочных файлов

```
#include <stdio.h>
#include "myfile.h"
```

Определение значений величин

```
#define N 10

int main() {
  int mas[N], i = 0;

  for (i = 0; i < N; i += 1)
    mas[i] = i;

  return 0;
}
```

Условная компиляция

```
#define N 10

#define DEBUG_PRINT

int main() {
  int mas[N], i = 0;

  for (i = 0; i < N; i += 1)
    mas[i] = i;

#if defined(DEBUG_PRINT)
  for (i = 0; i < N; i += 1)
    printf("%d\n", mas[i]);
#endif

  return 0;
}
```

Директивы препроцессора (2)

```
#define EXIT_FUNC { fclose(f); return 0; }

int main() {
  FILE *f = fopen("test.txt", "rt");
  if (!f)
    return 0;

  while (!feof(f)) {
    fscanf(f, ...
      if (...)
        EXIT_FUNC;
    ...
      if (...)
        EXIT_FUNC;
  }
  return 0;
}
```

```
#define EXIT_FUNC { fclose(f); return 0; }
```

```
#define EXIT_FUNC \
{ \
  fclose(f); \
  return 0; \
}
```

Ловим ошибки при считывании из файла

Ловим ошибки при обработке данных

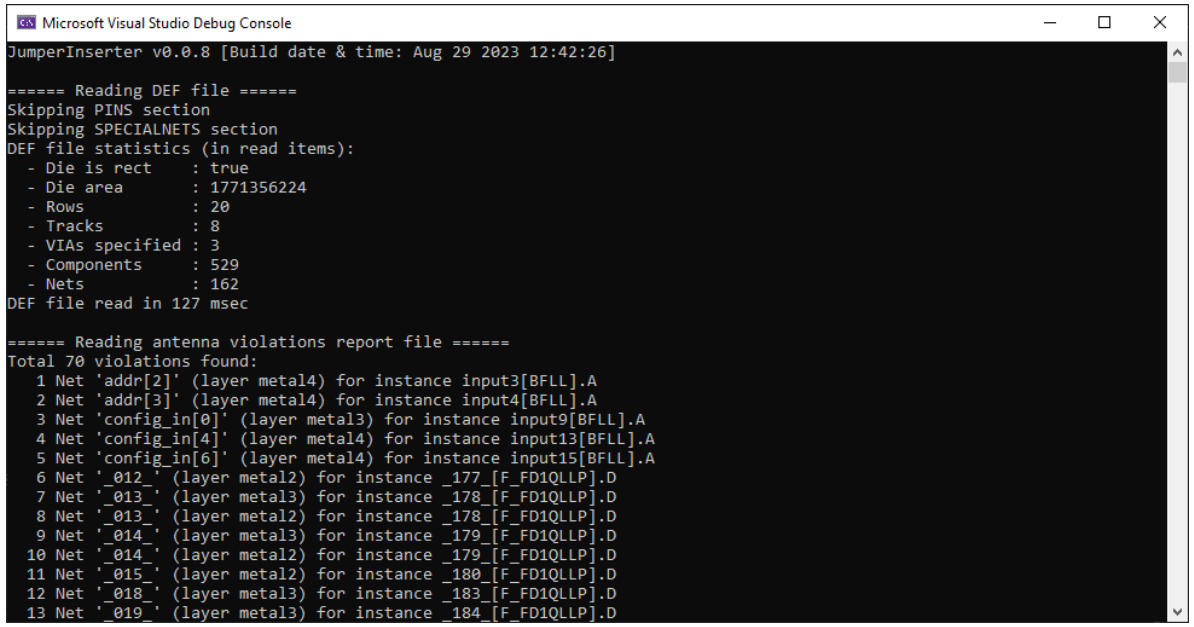
Директивы препроцессора (3)

```c
#include <stdio.h>

#define max(a, b) a > b ? a : b

int main() {
  int a = 0, b = 0;
  scanf("%d %d", &a, &b);

  printf("%d", max(a, b));

  return 0;
}
```

Директивы, объявляемые в момент компиляции

__TIME__ - вставляет время компиляции

__DATE__ - вставляет дату компиляции




```cpp
#include <iostream>

using namespace std;

int main() {
  cout << "Compile date is " << __DATE__ << endl;
  cout << "Compile time is " << __TIME__ << endl;
  return 0;
}
```

Compile date is Sep 02 2023
Compile time is 20:02:13

Директивы препроцессора: управление компиляцией (1)

Макрос определяется всеми Windows-компиляторами по общему соглашению

```
#if defined (_WIN32)
#error This program can't be compiled under MS Windows!"
#endif
```

main.cpp(5) : fatal error C1189: #error :
"This program can't be compiled under MS Windows!"

Директивы препроцессора: управление компиляцией (2)

```
int main() {
  float f = 0.45;
  return 0;
}
```

main.cpp(2) : warning C4305: 'initializing' : truncation from 'double' to 'float'

```
#pragma warning(disable: 4305)
```



```
#pragma warning(error: 4305)
```

```
int main() {
  float f = 0.45;
  return 0;
}
```

Директивы процессора: библиотека OpenMP

```
#include <stdio.h>
#include <omp.h>

#define SIZE 1000

int main() {
  printf("%d\n", omp_get_num_procs());

  omp_set_num_threads(4);

  int i = 0;
  int a[SIZE] = { 0 }, b[SIZE] = { 0 }, c[SIZE] = { 0 };

#pragma omp parallel
  {
#pragma omp for
    for (i = 0; i < SIZE; ++i)
      c[i] = a[i] + b[i];
  }
  return 0;
}
```

Что делает компилятор

«Догенерация» программного кода

```
class SomeClass {
public:
  SomeClass() {
  }
  ~SomeClass() {
  }
};
```

Класс (конструктор и деструктор)

```
template <typename T>
T test_abs(T arg) {
  if (arg > 0)
    return arg;
  return (-arg);
}
```

Шаблоны

```
int test_arg(int a, int b = 0) {
  return a * b;
}
```

Аргументы функций по умолчанию

```
int main() {
  SomeClass sc;

  int x = test_abs(-42);
  double y = test_abs(-42.0);

  int z = test_arg(x);

  return 0;
}
```

Используем класс

Вызываем шаблонные функции с разными типами аргументов

Вызываем функцию с одним аргументом

Лексический анализ

```
for (i = 0; i < max; ++i)
  printf("Hello\n");
```

```
01 : for
02 : (
03 : i
04 : =
05 : 0
06 : ;
07 : i
08 : <
09 : max
10 : ;
11 : +
12 : +
13 : i
14 : )
15 : printf
16 : (
17 : "
18 : Hello
19 : \
20 : n
21 : "
22 : )
23 : ;
```

```
11 : +
12 : +
```

```
11 : ++
```

```
19 : \
20 : n
```

```
19 : \n
```

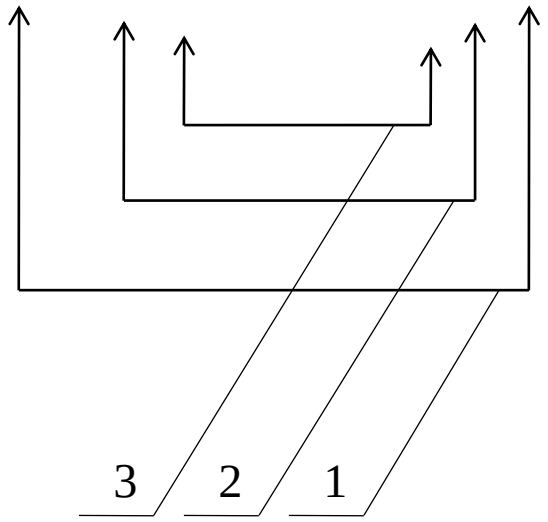
Синтаксический анализ

```
for (i = 0; i < max; ++i)
    printf("Hello\n");
```

```
for ( A ; B ; C )
```

```
printf ( D ) ;
printf ( " Hello\n " ) ;
```

```
210  #include <iostream>
211  #include <vector>
212  using namespace std;
213  int main() {
214      for(int i = 0, i < 10, i++)
215
216      return 0;
217  }
```




```
1>main.cpp
1>main.cpp(214,20): error C2143: syntax error: missing ',' before '<'
1>main.cpp(214,18): error C2086: 'int i': redefinition
1>main.cpp(214,11): message : see declaration of 'i'
1>Done building project "test_Console.vcxproj" -- FAILED.
```

Семантический анализ

```
int i = 0;
unsigned int max = 10;
for (i = 0; i < max; ++i)
```

```
main.cpp(7) : warning C4018: '<' : signed/unsigned mismatch
```

```
double a = 1.0;
int b = a;
```

```
main.cpp(6) : warning C4244: 'initializing' : conversion from 'double' to 'int',
possible loss of data
```

```
int a = sqrt(4);
```

```
main.cpp(6) : error C2668: 'sqrt' : ambiguous call to overloaded function

1> ..\include\math.h(581): could be 'long double sqrt(long double)'
1> ..\include\math.h(533): or           'float sqrt(float)'
1> ..\include\math.h(128): or           'double sqrt(double)'
```

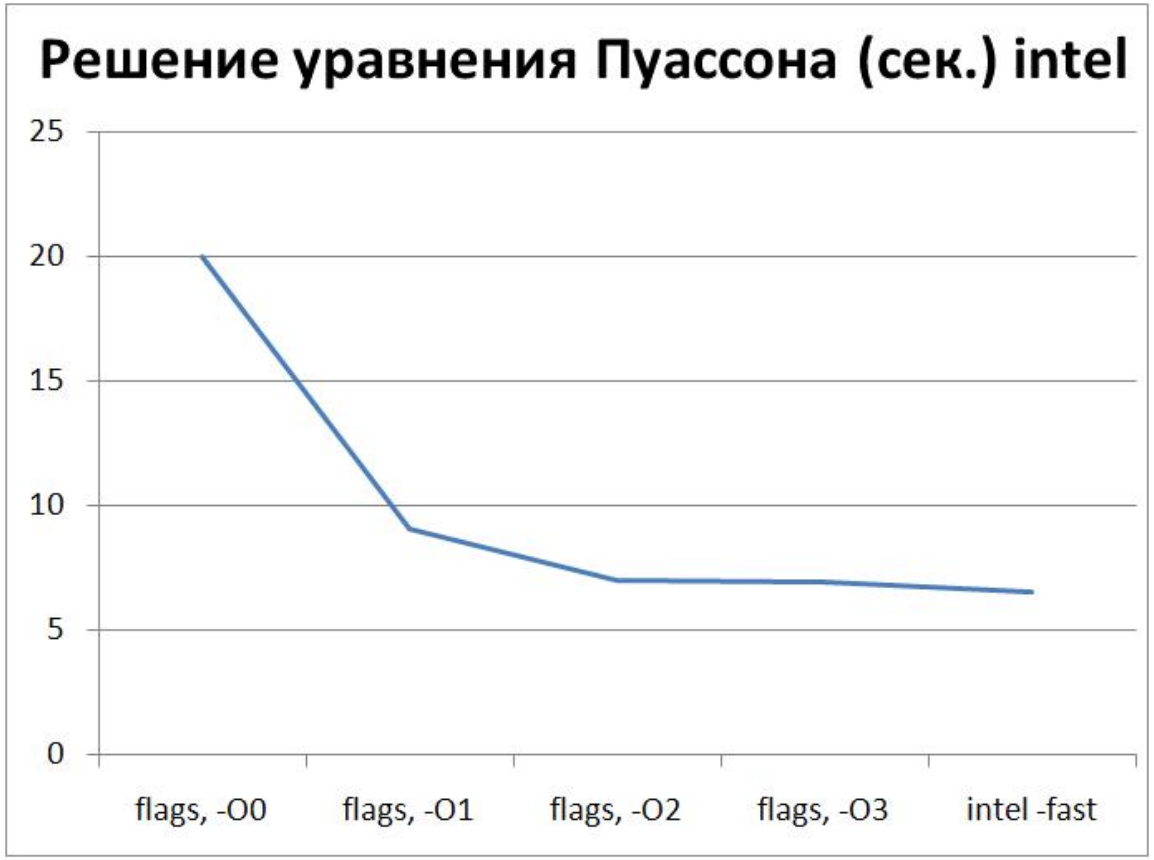
Оптимизация программного кода

Оптимизация выполняется для увеличения производительности

Примеры оптимизаций кода:

1. оптимизация программных инструкций – замена, переупорядочивание;
2. понижение силы операций;
3. оптимизация сложных множественных ветвлений;
4. встраивание inline функций;
5. удаление неиспользуемого кода;
6. выбор регистров;
7. …

Основные ключи оптимизации компиляторов C++



-O0 – оптимизации нет

-O1 – увеличить скорость

-O2 – увеличить скорость, увеличив при этом объём кода

-O3 – самая жёсткая оптимизация

Пример оптимизации: понижение силы операций

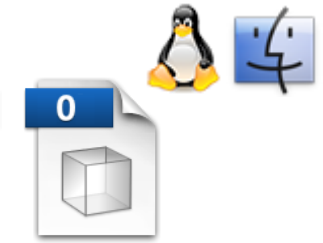
```
// до оптимизации (3 такта на Core 2 Duo)

y = 2 * x;


// после оптимизации

y = x + x;  // 1 такт на Core 2 Duo

или

y = x << 1; // 1 такт на Core 2 Duo
```

Процесс сборки программ

Работа компилятора

Предварительная обработка исходного кода препроцессором

Компиляция исходного кода каждого отдельного файла



Работа линковщика

Линковка нескольких объектных модулей в один файл

com exe	не имеет .out
dll	so

Что делает компоновщик?

Файл 1.cpp

Локальные для файла 1 переменные

Файл 2.cpp

Локальные для файла 2 переменные

Файл 3.cpp

Локальные для файла 3 переменные

Проверка того, что имена типов данных и переменных не пересекаются

Проверка того, что тела всех вызываемых функций найдены

Исполняемый модуль

Пример ошибки компоновщика

Файл main.cpp:

```
int a = 6;

int main() {
  int b = 4 + a;
}
```

Файл func.cpp:

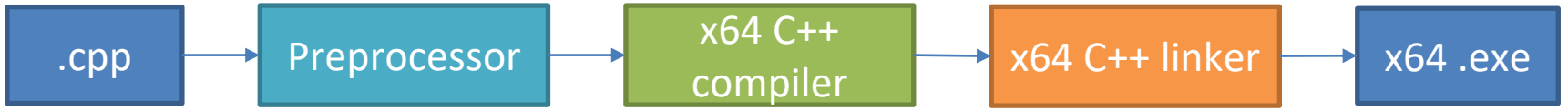
```
int a = 4;

int func() {
  int c = 2 * a;
}
```

```
1>main2.obj : error LNK2005: "int a" (?a@@3HA) already defined in
main.obj
```

Примерная архитектура работы обычных компиляторов

Windows, x86

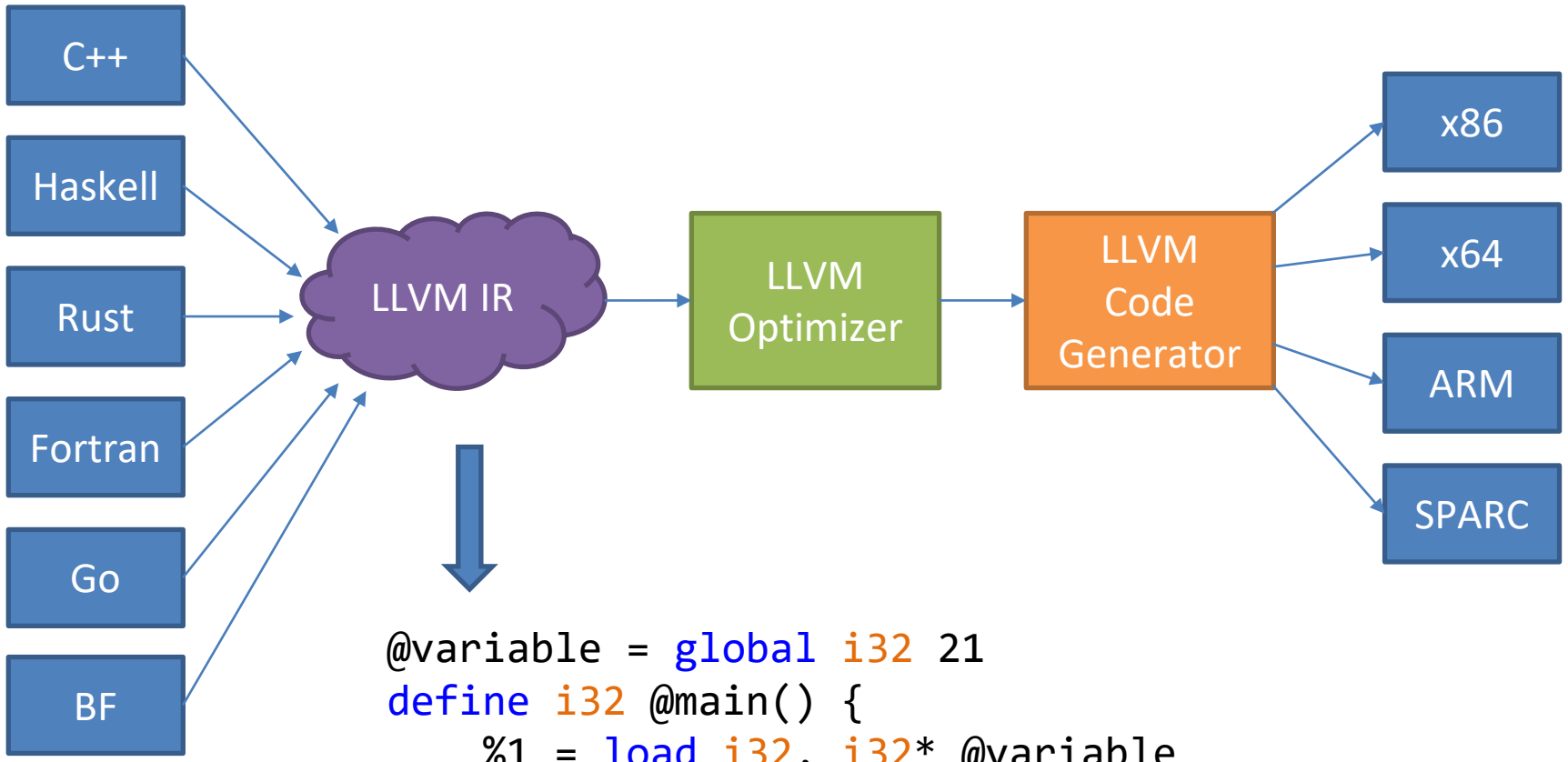
Windows, x64



Linux, gcc

Fortran

LLVM: Low Level Virtual Machine (1)




```
@variable = global i32 21
define i32 @main() {
    %1 = load i32, i32* @variable
    %2 = mul i32 %1, 2
    store i32 %2, i32* @variable
    ret i32 %2
}
```

LLVM: Low Level Virtual Machine (2)

Типы данных:

- целые типы данных произвольной разрядности:
  `i1`, `i5`, `i32`, `i61`, `i64`
- числа с плавающей точкой:
  `float`, `double`
- пустое значение:
  `void`
- указатели:
  `i32*`, `i17*`

и много чего ещё...

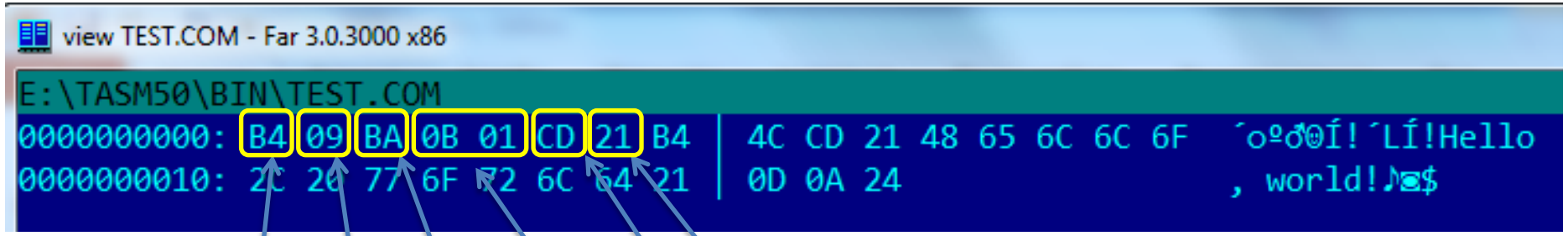
Глобальные переменные обозначаются префиксом @

Локальные переменные обозначаются префиксом %

Можно использовать функции и много чего ещё...

```
@variable = global i32 21
define i32 @main() {
    %1 = load i32, i32* @variable
    %2 = mul i32 %1, 2
    store i32 %2, i32* @variable
    ret i32 %2
}
```

Как выполняется программа на уровне МП?




```
.model tiny
.code
org 100h
main:
        mov AH, 09h
        lea DX, string
        int 21h

        mov AH, 4Ch
        mov AL, 00h
        int 21h

        string db "Hello, world!", 0Dh, 0Ah, '$'

end main
```