```
#include<stdio.h>
int main() {

   printf("Hello World!");
   return 0;
}
```

Compiler Design

Introduction

Punched Card:

ASCII

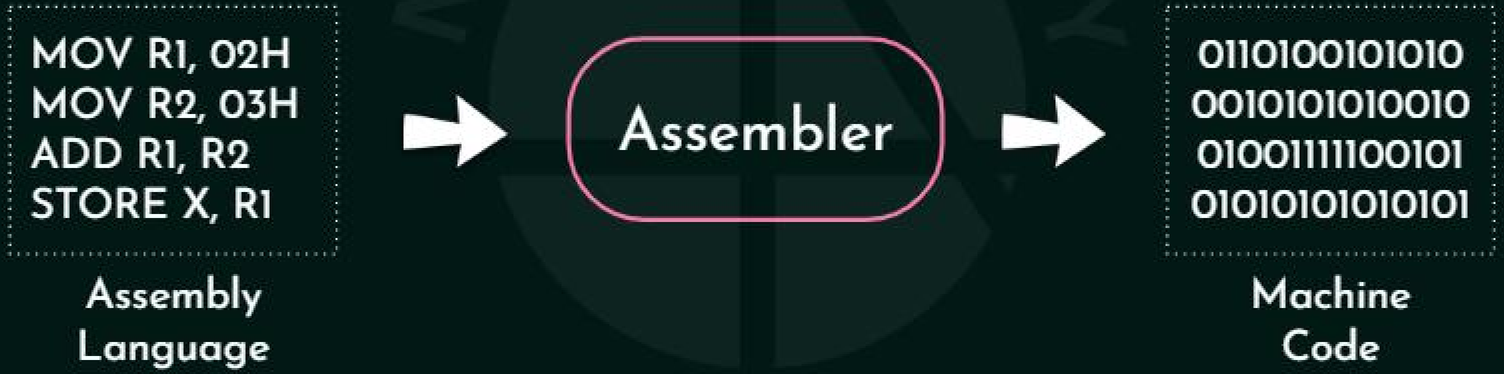
P –	1010000
u –	1110101
n –	1101110
c –	1100011
h –	1101000
e –	1100101
d –	1100100

C –	1000011
a –	1100001
r –	1110010
d –	1110011

Language Translator:

Language Translator

i. Assembler:



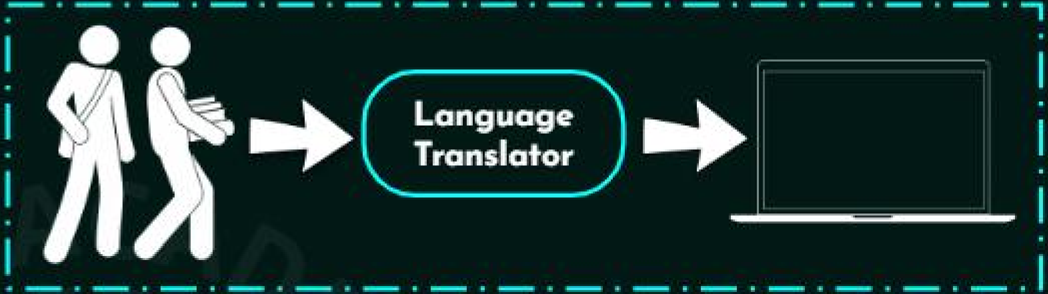
```
MOV R1, 02H
MOV R2, 03H
ADD R1, R2
STORE X, R1
```

Assembly Language

```
0110100101010
0010101010010
0100111100101
01010101010101
```

Machine Code

Language Translator:



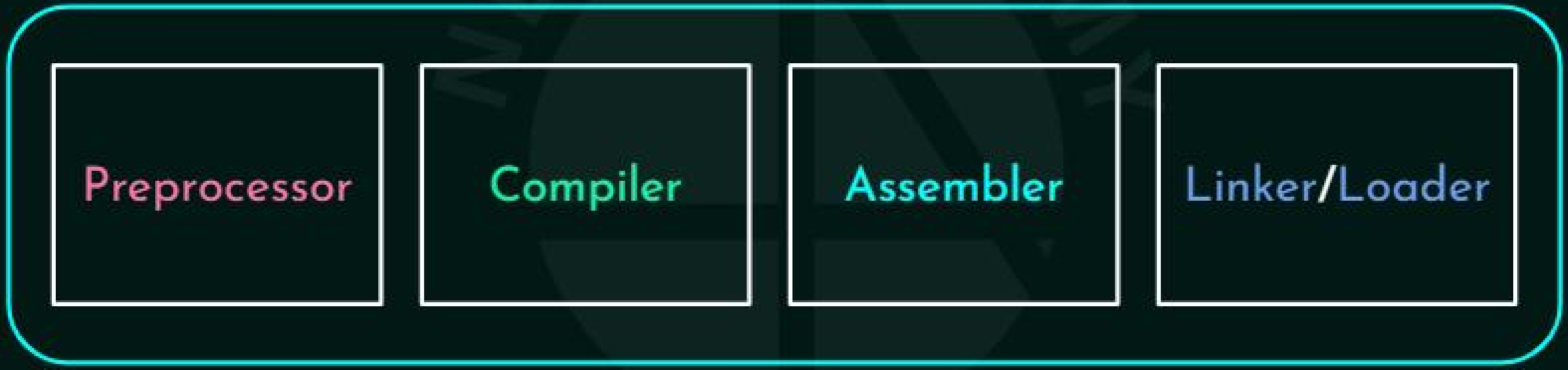
i. Assembler

ii. Interpreter:

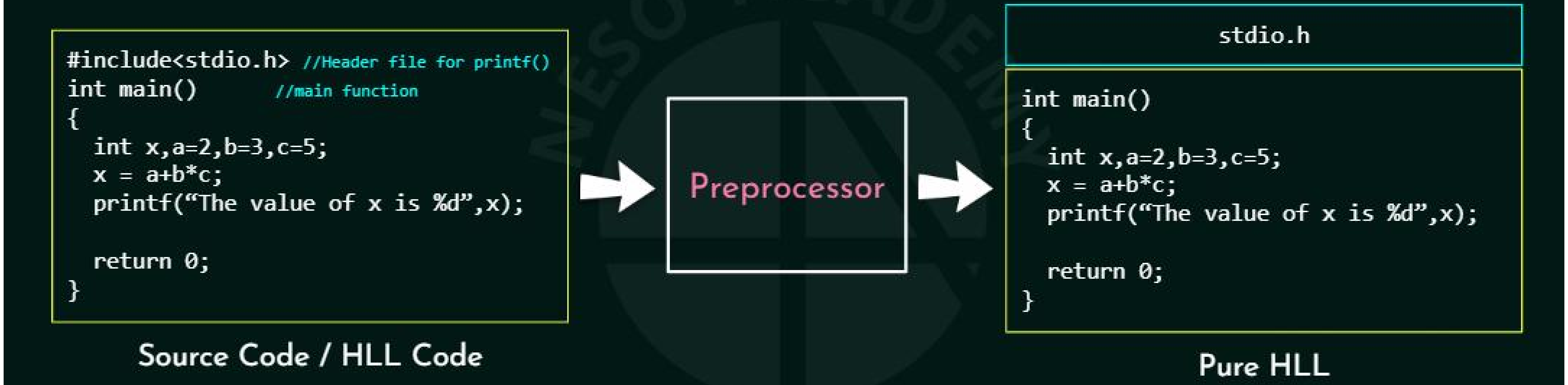
php
JAVASCRIPT
JS
Ruby

iii. Compiler:

Language Translator – Internal Architecture



Language Translator – Internal Architecture



```
#include<stdio.h> //Header file for printf()
int main()        //main function
{
  int x,a=2,b=3,c=5;
  x = a+b*c;
  printf("The value of x is %d",x);

  return 0;
}
```

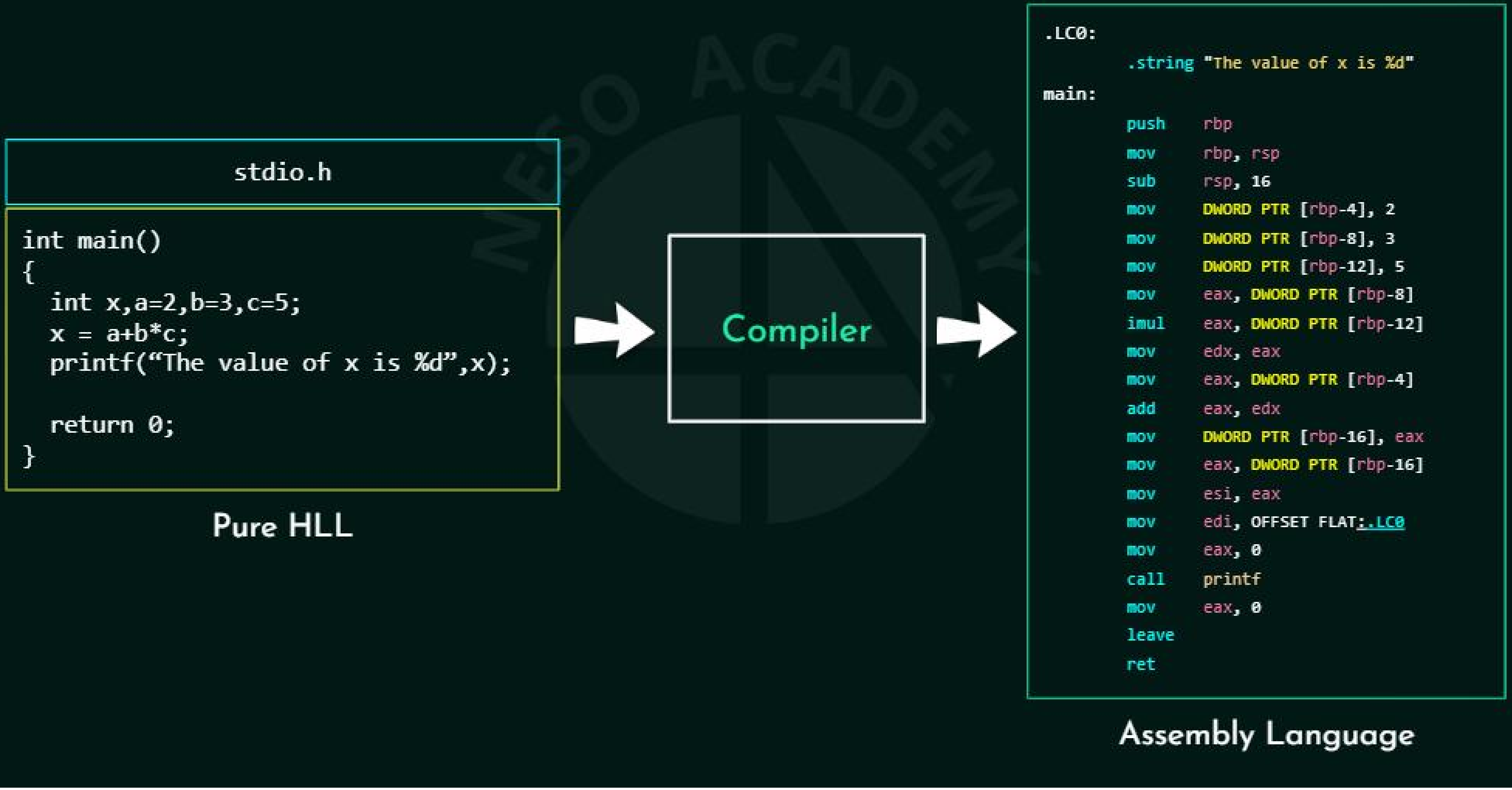
Source Code / HLL Code

```
stdio.h
```

```
int main()
{
  int x,a=2,b=3,c=5;
  x = a+b*c;
  printf("The value of x is %d",x);

  return 0;
}
```

Pure HLL

Language Translator – Internal Architecture



```
stdio.h
```

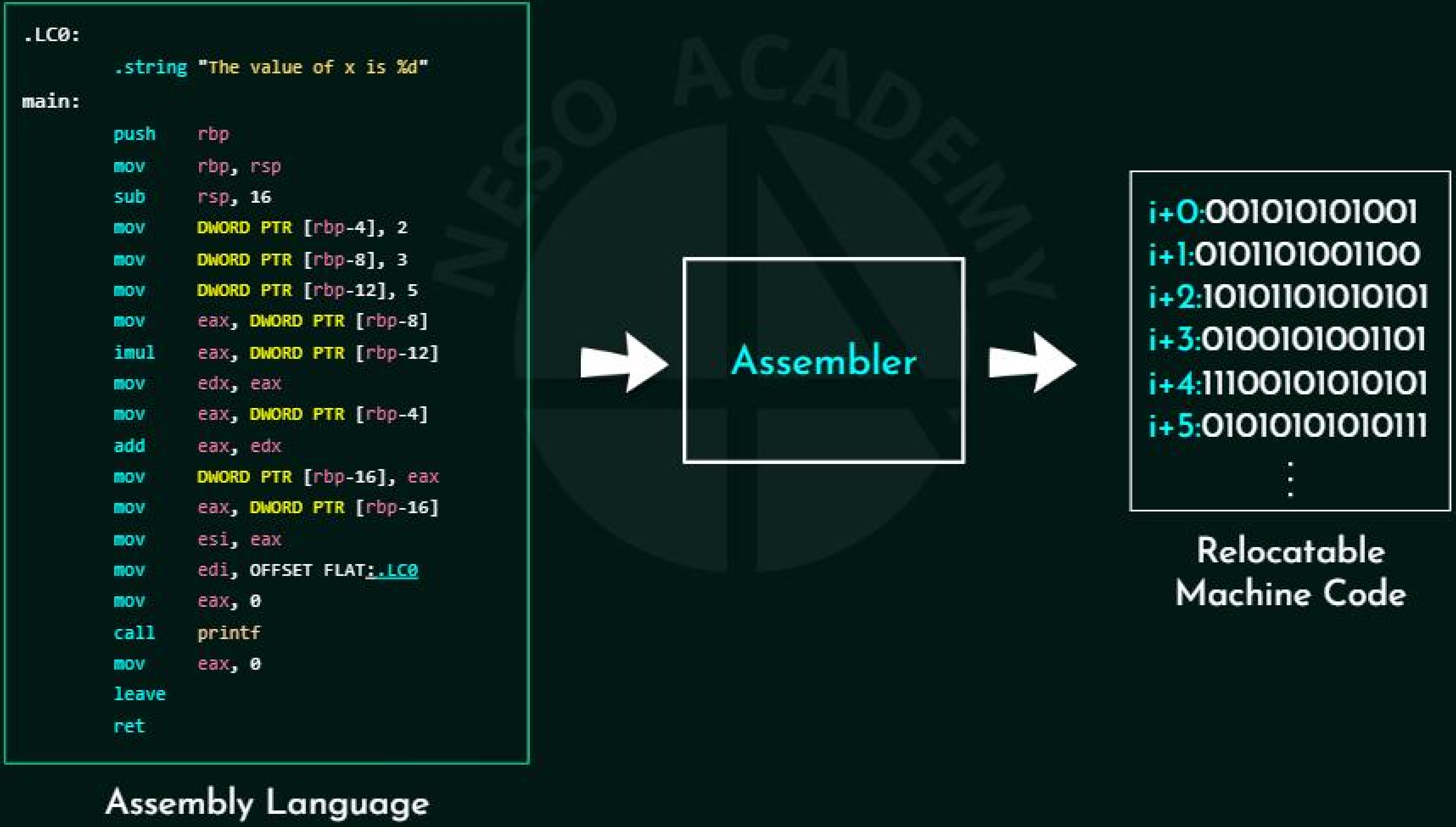
```
int main()
{
  int x,a=2,b=3,c=5;
  x = a+b*c;
  printf("The value of x is %d",x);

  return 0;
}
```

Pure HLL

```
.LC0:
        .string "The value of x is %d"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 2
        mov     DWORD PTR [rbp-8], 3
        mov     DWORD PTR [rbp-12], 5
        mov     eax, DWORD PTR [rbp-8]
        imul    eax, DWORD PTR [rbp-12]
        mov     edx, eax
        mov     eax, DWORD PTR [rbp-4]
        add     eax, edx
        mov     DWORD PTR [rbp-16], eax
        mov     eax, DWORD PTR [rbp-16]
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    printf
        mov     eax, 0
        leave
        ret
```

Assembly Language

Language Translator – Internal Architecture



```
.LC0:
        .string "The value of x is %d"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 2
        mov     DWORD PTR [rbp-8], 3
        mov     DWORD PTR [rbp-12], 5
        mov     eax, DWORD PTR [rbp-8]
        imul    eax, DWORD PTR [rbp-12]
        mov     edx, eax
        mov     eax, DWORD PTR [rbp-4]
        add     eax, edx
        mov     DWORD PTR [rbp-16], eax
        mov     eax, DWORD PTR [rbp-16]
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    printf
        mov     eax, 0
        leave
        ret
```

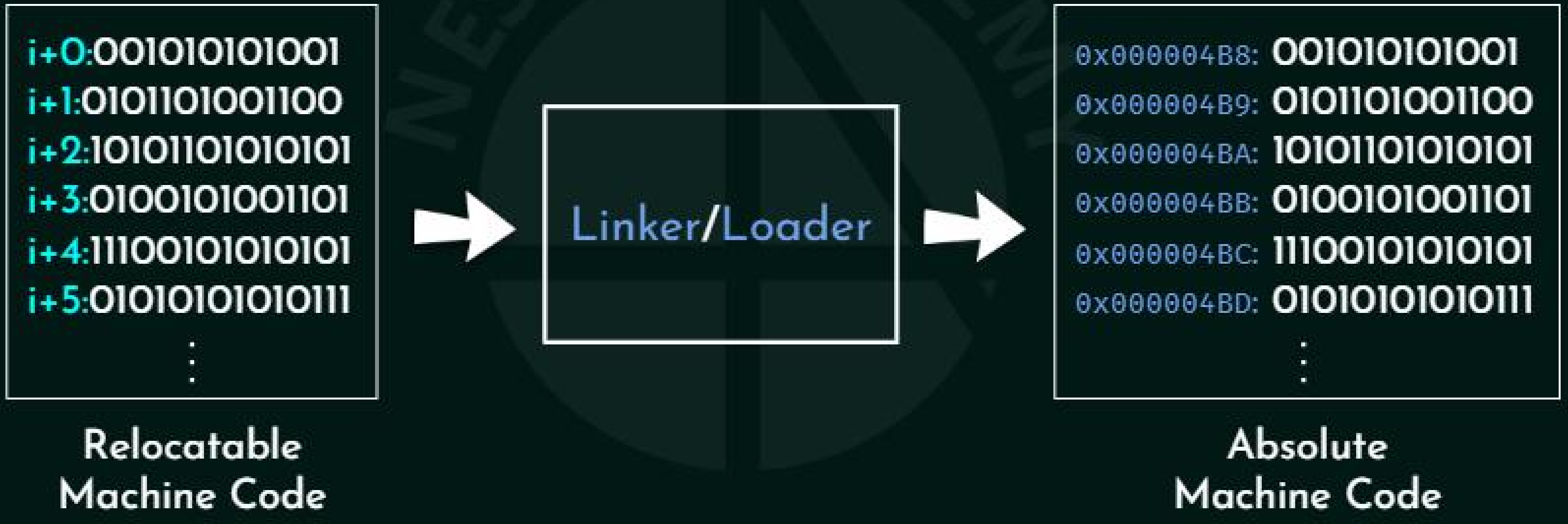
Assembly Language

Assembler

```
i+0:001010101001
i+1:0101101001100
i+2:10101101010101
i+3:0100101001101
i+4:11100101010101
i+5:01010101010111
      ⋮
```

Relocatable
Machine Code

Language Translator – Internal Architecture



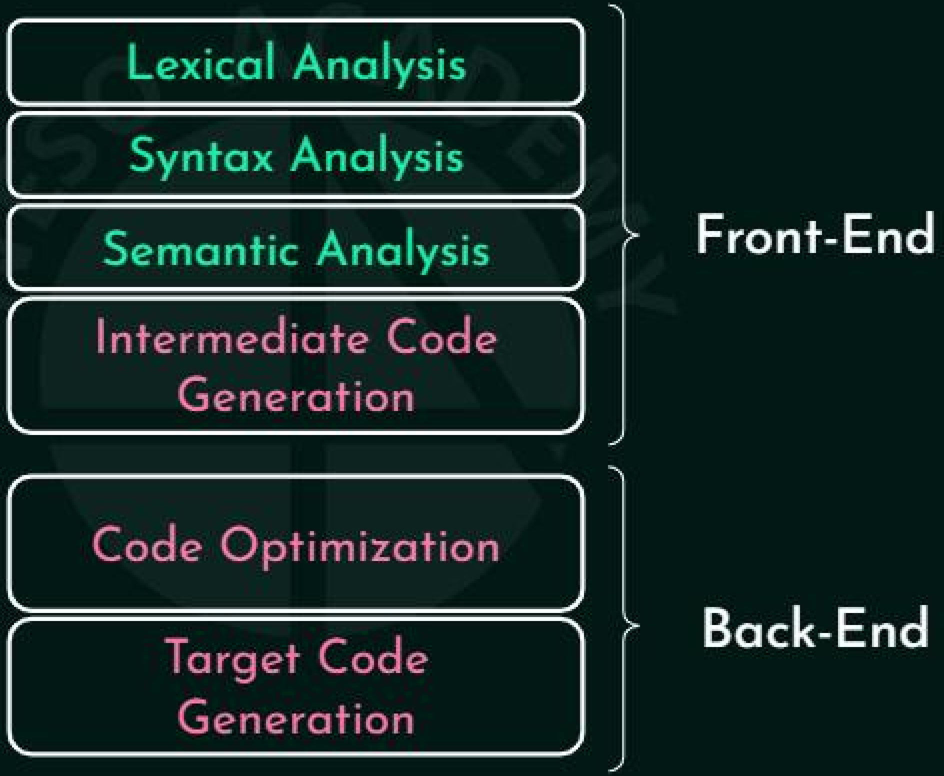
```
i+0:001010101001
i+1:0101101001100
i+2:10101101010101
i+3:0100101001101
i+4:11100101010101
i+5:01010101010111
⋮
```

Relocatable
Machine Code

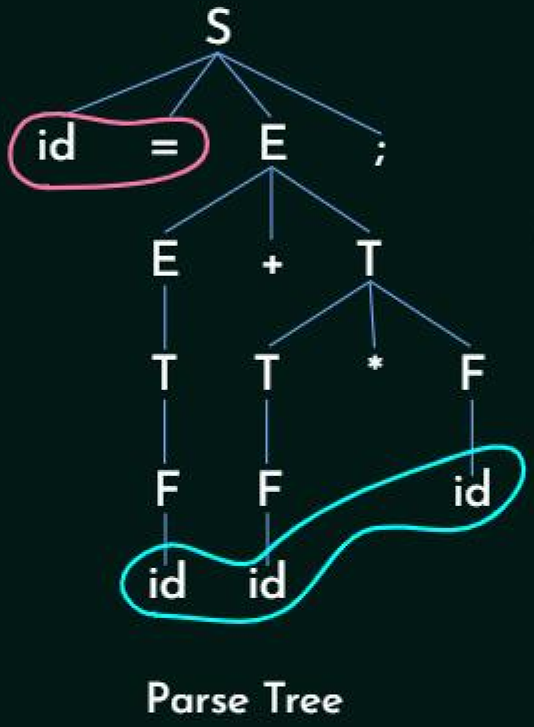
```
0x000004B8: 001010101001
0x000004B9: 0101101001100
0x000004BA: 10101101010101
0x000004BB: 0100101001101
0x000004BC: 11100101010101
0x000004BD: 01010101010111
⋮
```

Absolute
Machine Code

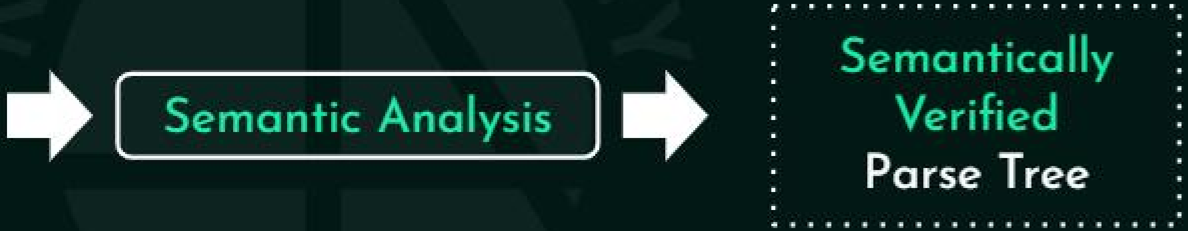
Compiler – Internal Architecture



Semantic Analyzer:

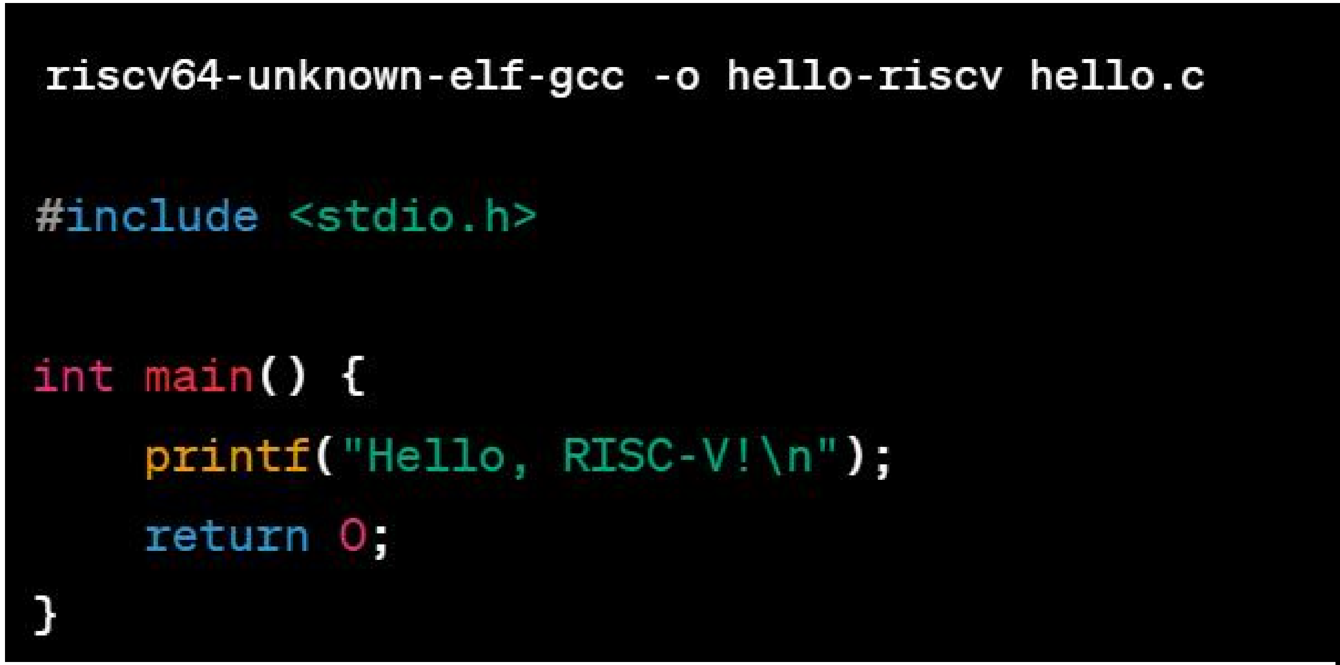


Parse Tree

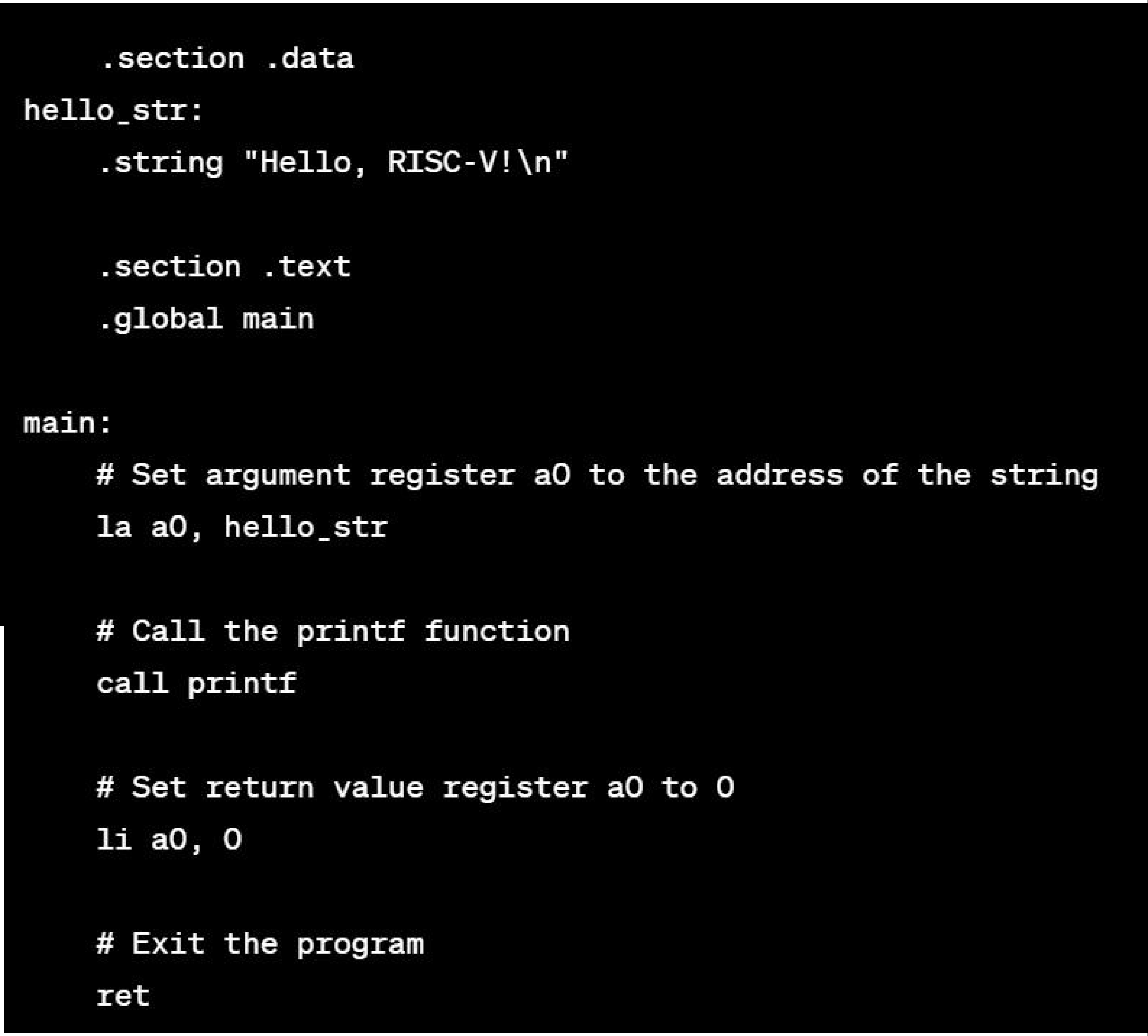


Semantic Analysis

Semantically Verified Parse Tree

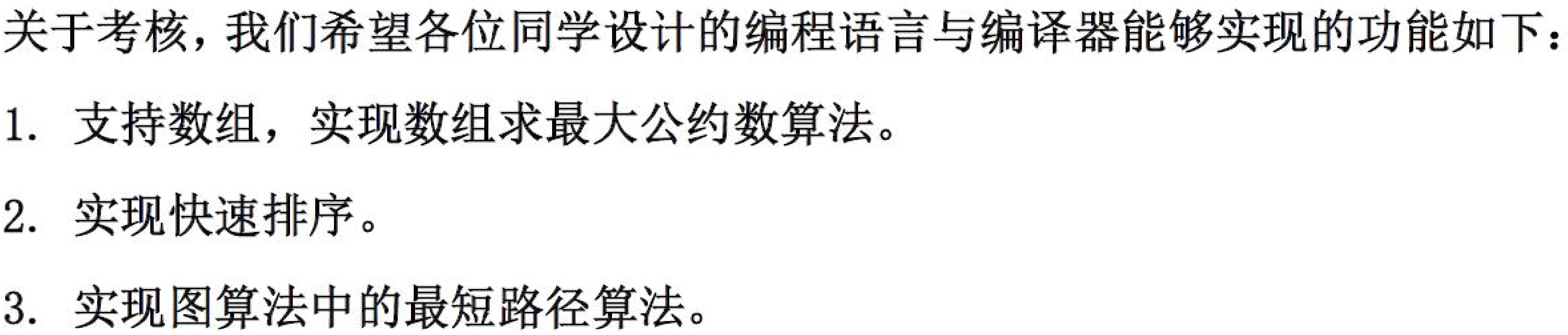


```
riscv64-unknown-elf-gcc -o hello-riscv hello.c

#include <stdio.h>

int main() {
    printf("Hello, RISC-V!\n");
    return 0;
}
```



```
    .section .data
hello_str:
    .string "Hello, RISC-V!\n"

    .section .text
    .global main

main:
    # Set argument register a0 to the address of the string
    la a0, hello_str

    # Call the printf function
    call printf

    # Set return value register a0 to 0
    li a0, 0

    # Exit the program
    ret
```

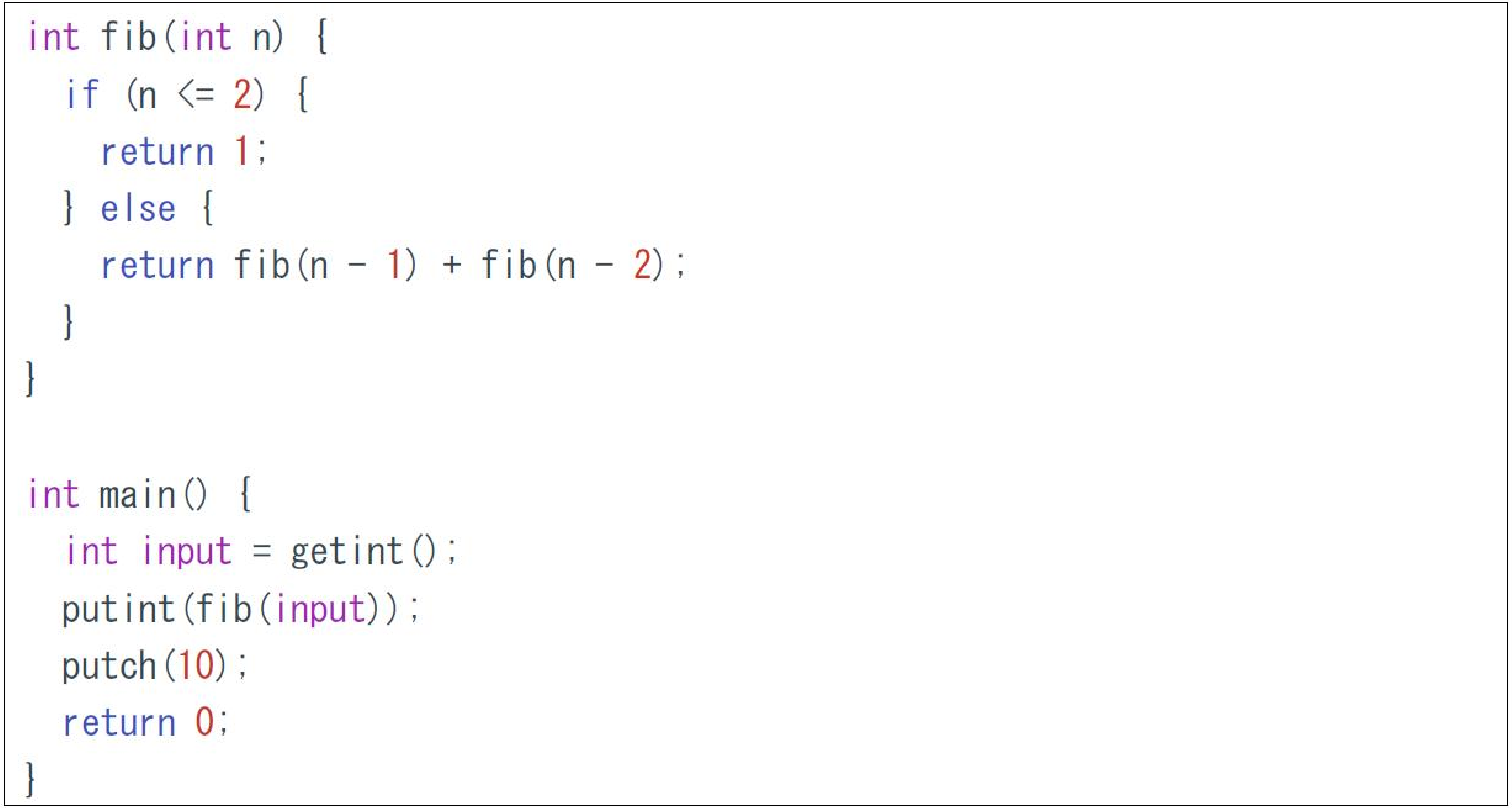
一期编译器部分实验目标



关于考核，我们希望各位同学设计的编程语言与编译器能够实现的功能如下：

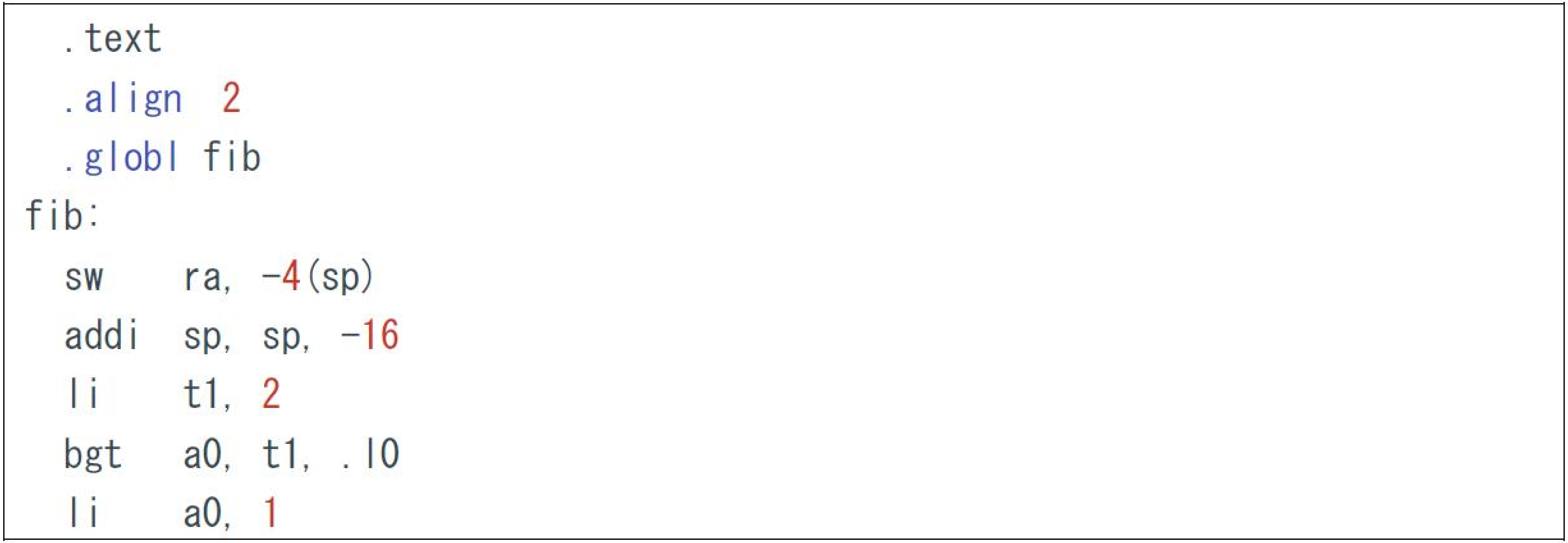
1. 支持数组，实现数组求最大公约数算法。
2. 实现快速排序。
3. 实现图算法中的最短路径算法。

一期编译器部分实验目标



```
int fib(int n) {
  if (n <= 2) {
    return 1;
  } else {
    return fib(n - 1) + fib(n - 2);
  }
}

int main() {
  int input = getint();
  putint(fib(input));
  putch(10);
  return 0;
}
```

一期编译器部分实验目标



```
   .text
   .align  2
   .globl fib
fib:
   sw     ra, -4(sp)
   addi   sp, sp, -16
   li     t1, 2
   bgt    a0, t1, .l0
   li     a0, 1
```

二期编译器部分实验目标（选做）

1. 实现一个类C的编程语言，并通过接入llvm实现RISC V的汇编代码生成

2. 通过gcc，或者clang+llvm，将c程序交叉编译成riscv代码在处理器上运行，并验证浮点指令等。