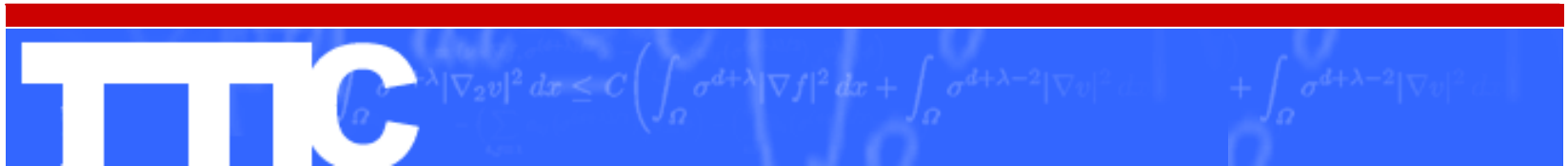


Introduction to C-- as an alternative backend



2007.5.1

Wonseok Chae

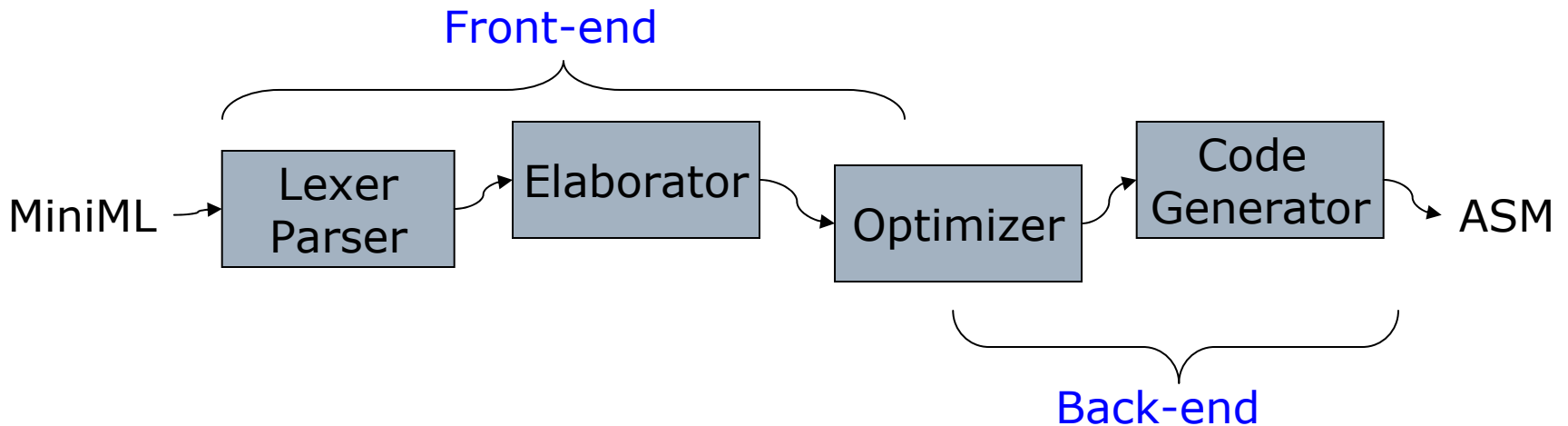
TTI-C

Outline

- Motivation
- Introduction to C--
- The MLPolyR frontend compiler
- Summary

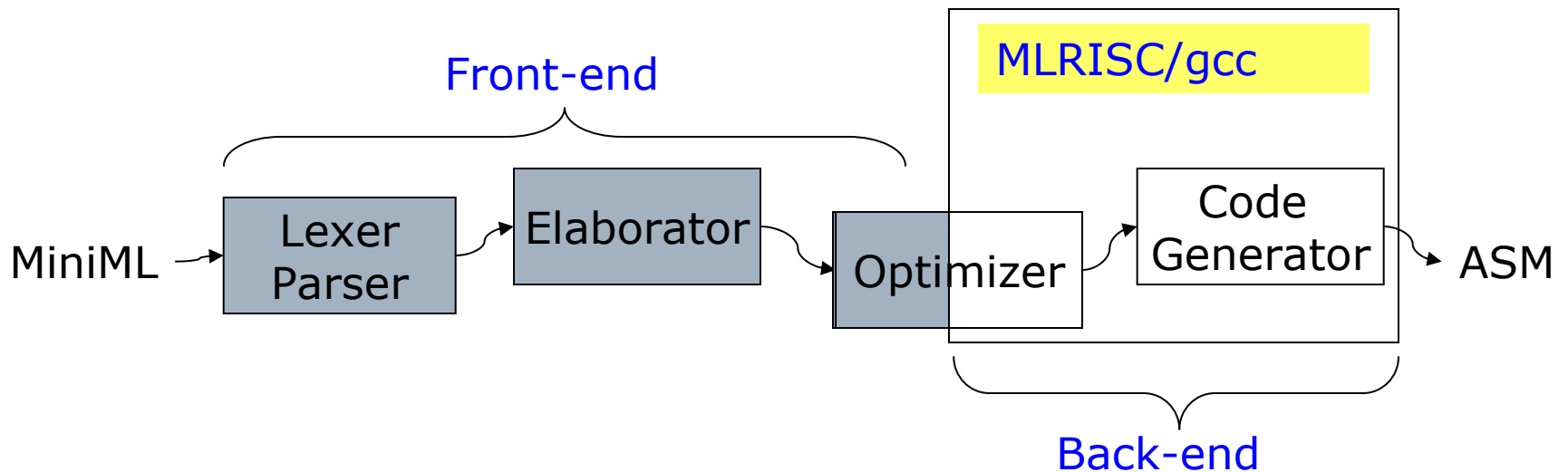
Announcement

- The project 3 is to write a code generator for MiniML to produce PowerPC assembler codes. FYI, you have only two days left :-)



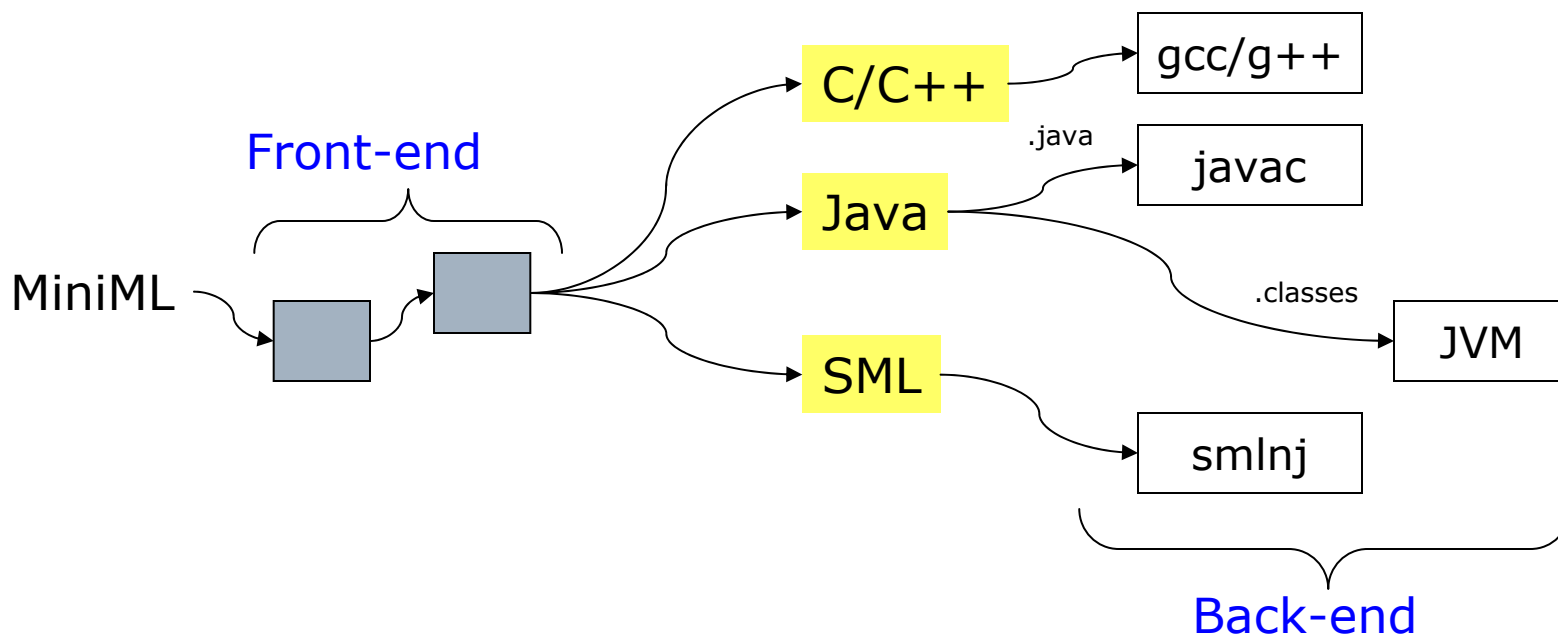
In the real world

- You may try to take advantage of an off-the-shelf code generator such as MLRISC or the gcc backend.



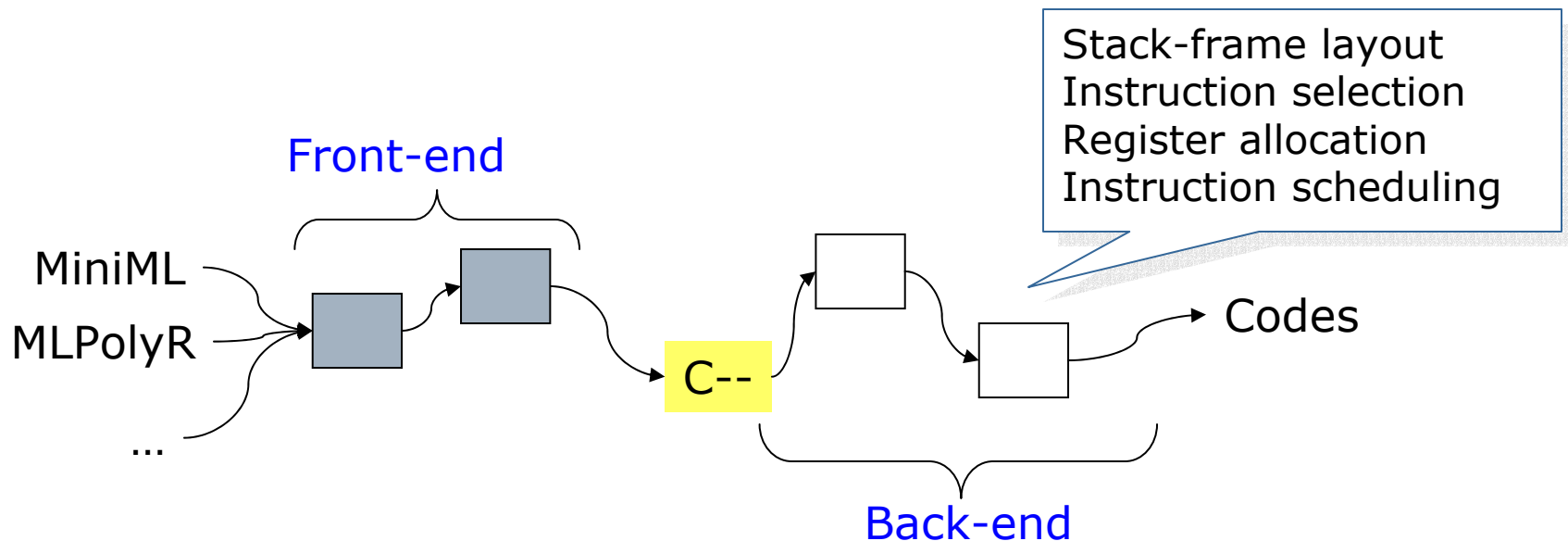
The most popular choice

- Have you heard of a portable assembly language, called "C"?



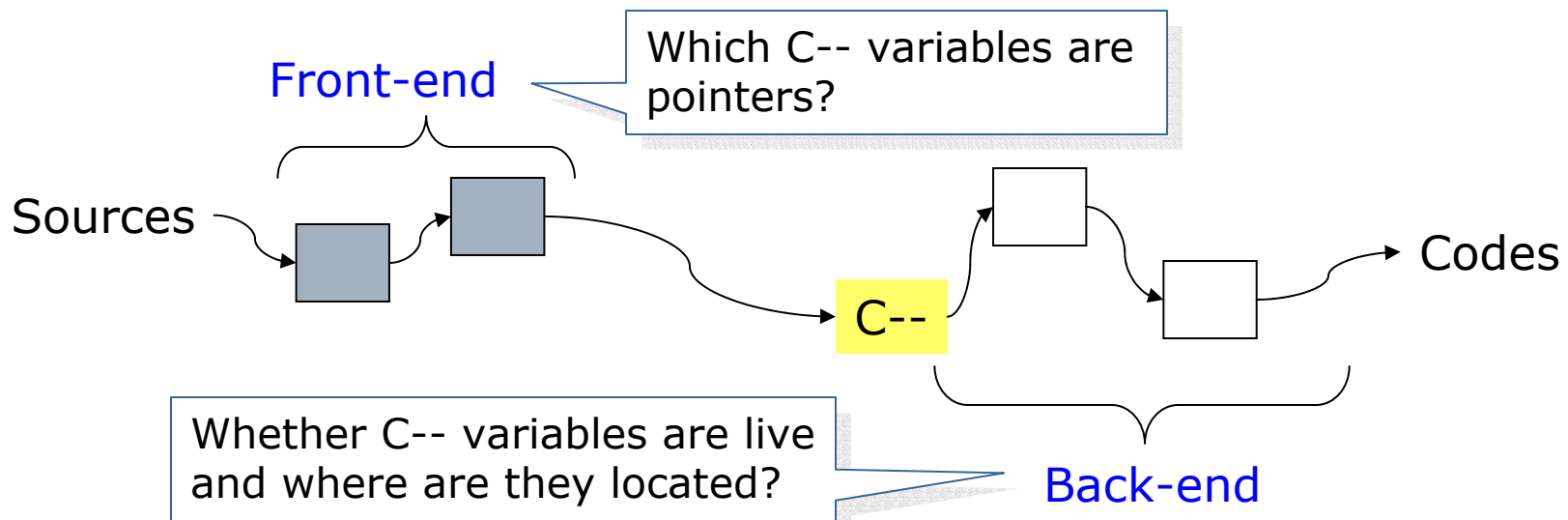
A portable assembly language

- It is an interface between a high-level compiler and a code generator.



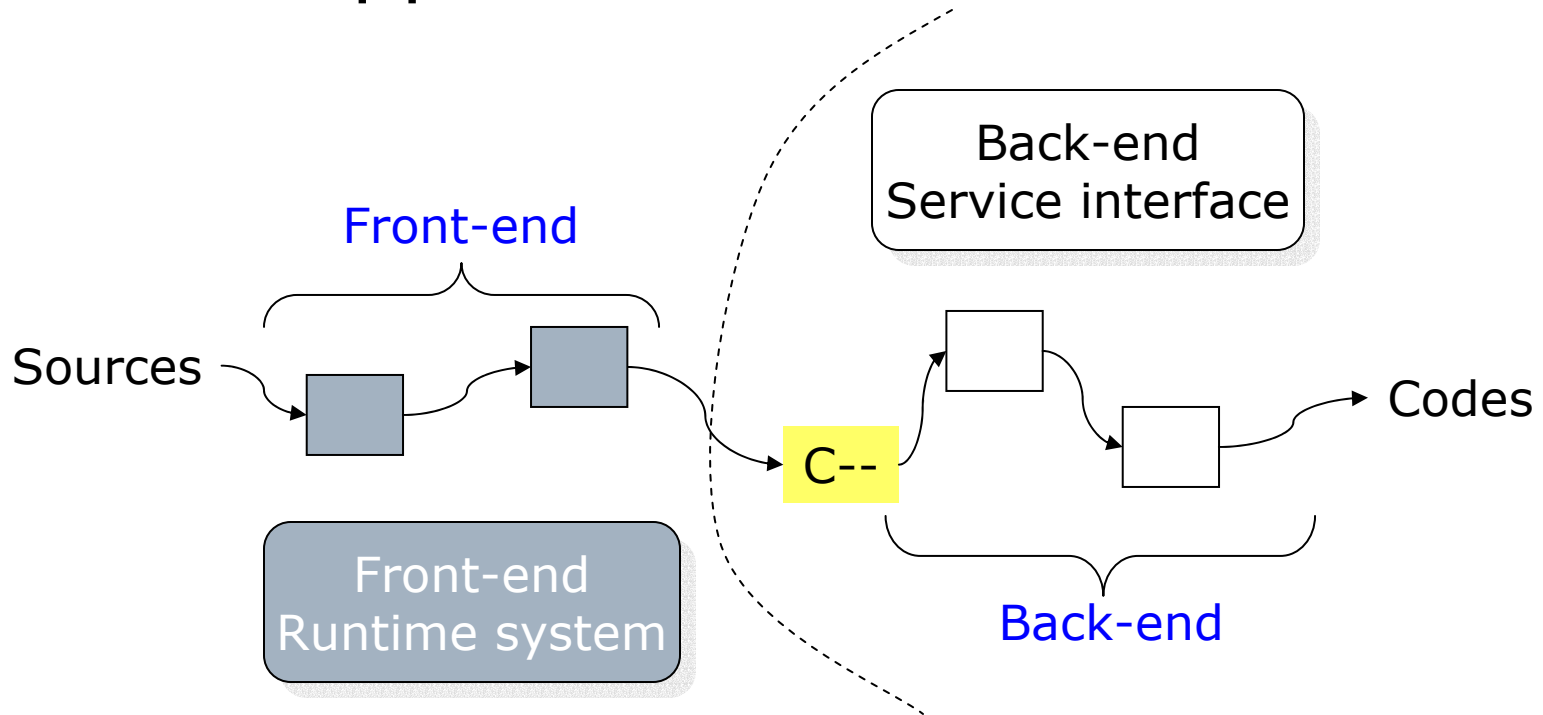
...causes a new problem

- Separating the front and back ends complicates run-time supports such as garbage collection.



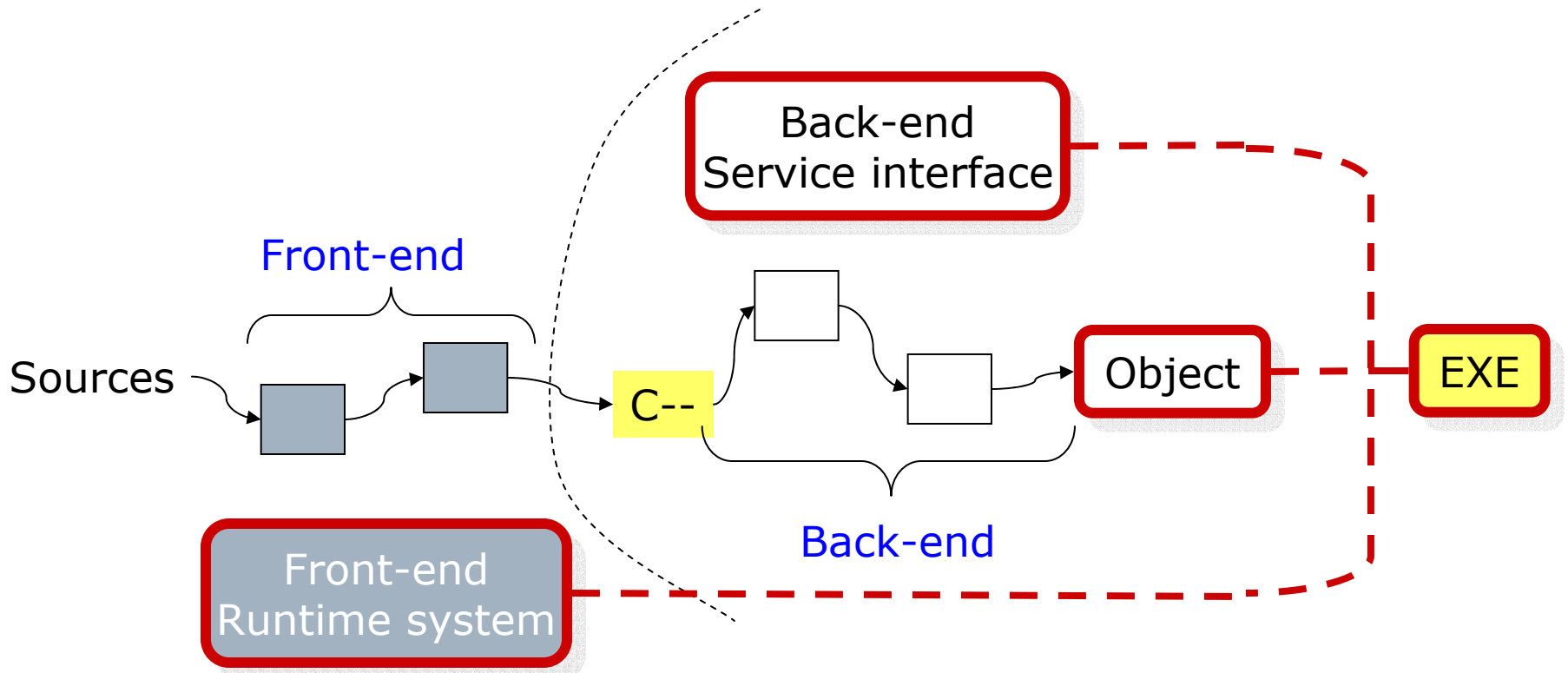
The C-- framework (1)

- C-- is a portable assembler language that supports run-time services.



The C-- framework (2)

- An executable program is ...



A language feature of C-- (1)

□ Types

- bits8, bits16, bits32, bits64
- float32, float64, float80

Types are only for the back end to choose the proper machine instruction for each operation, depending on the size of the operands.

A language feature of C-- (2)

□ Static memory layout

```
section "data" { align 4;
  l_mlpr_string_2: bits8[] "\n\000";
  l_mlpr_string_1: bits8[] ")=\000";
  l_mlpr_string_0: bits8[] "ack(3,\000";
  foo: bits32{17};
}
```

(a) C--

```
.data
.cstring
    .align 2
l_mlpr_string_0:
    .asciz "ack(3,"
```

(b) PowerPC assembler

```
f() {
  /* memory read/write */
  bits32[foo] = bits32[foo] + 1;
  return ();
}
```

A language feature of C-- (3)

□ Static memory layout

```
section "data" { align 4;  
  label_0:bits8[] "Hello";  
  label_1:bits8[] "world\n";  
}
```

Q1: What is the value of label_0?

A language feature of C-- (4)

□ Static memory layout

```
section "data" { align 4;  
  label_0:bits8[] "Hello";  
  label_1:bits8[] "world\n";  
}
```

Q1: What is the value of label_0?
A1: "Helloworld\n...", not "Hello".

A language feature of C-- (5)

□ Static memory layout

```
section "data" { align 4;
  label_0:bits8[] "Hello";
  label_1:bits8[] "world\n";
}
```

Q1: What is the value of label_0?
A1: "Helloworld\n...", not "Hello".

Q2: If label_0 points to "0x1000",
where does label_1 point?

A language feature of C-- (6)

□ Static memory layout

```
section "data" { align 4;
  label_0:bits8[] "Hello";
  label_1:bits8[] "world\n";
}
```

Q1: What is the value of label_0?
A1: "Helloworld\n...", not "Hello".

Q2: If label_0 points to "0x1000",
where does label_1 point?
A1: "0x1005", not "0x1008".

```
section "data" {
  align 4; label_0:bits8[] "Hello\000";
  align 4; label_1:bits8[] "world\n\000";
}
```

A language feature of C-- (7)

□ Procedures

```
sp1 (bits32 n) {  
  bits32 s,p;  
  if n==1 {  
    return (1,1);  
  } else {  
    s, p = sp1 (n-1);  
    return (s+n, p*n);  
  }  
}
```

(a) Ordinary recursion

```
sp2 (bits32 n) {  
  bits32 s,p;  
  s = 1; p = 1;  
  loop:  
  if n==1 {  
    return (s,p);  
  } else {  
    s = s+n;  
    p = p*n;  
    n = n-1;  
    goto loop;  
  }  
}
```

(b) loops

```
sp3 (bits32 n) {  
  jump spj(n, 1, 1);  
}  
  
spj (bits32, bits32 s, bits32 p) {  
  if n==1 {  
    return (s, p);  
  } else {  
    jump spj (n-1, s+n, p*n);  
  }  
}
```

(c) tail recursion

A language feature of C-- (8)

□ Calling conventions

```
export sp1;  
  
sp1 (bits32 n) {  
  bits32 s,p;  
  if n==1 {  
    return (1,1);  
  } else {  
    s, p = sp1 (n-1);  
    return (s+n, p*n);  
  }  
}
```

(a) sp.c--

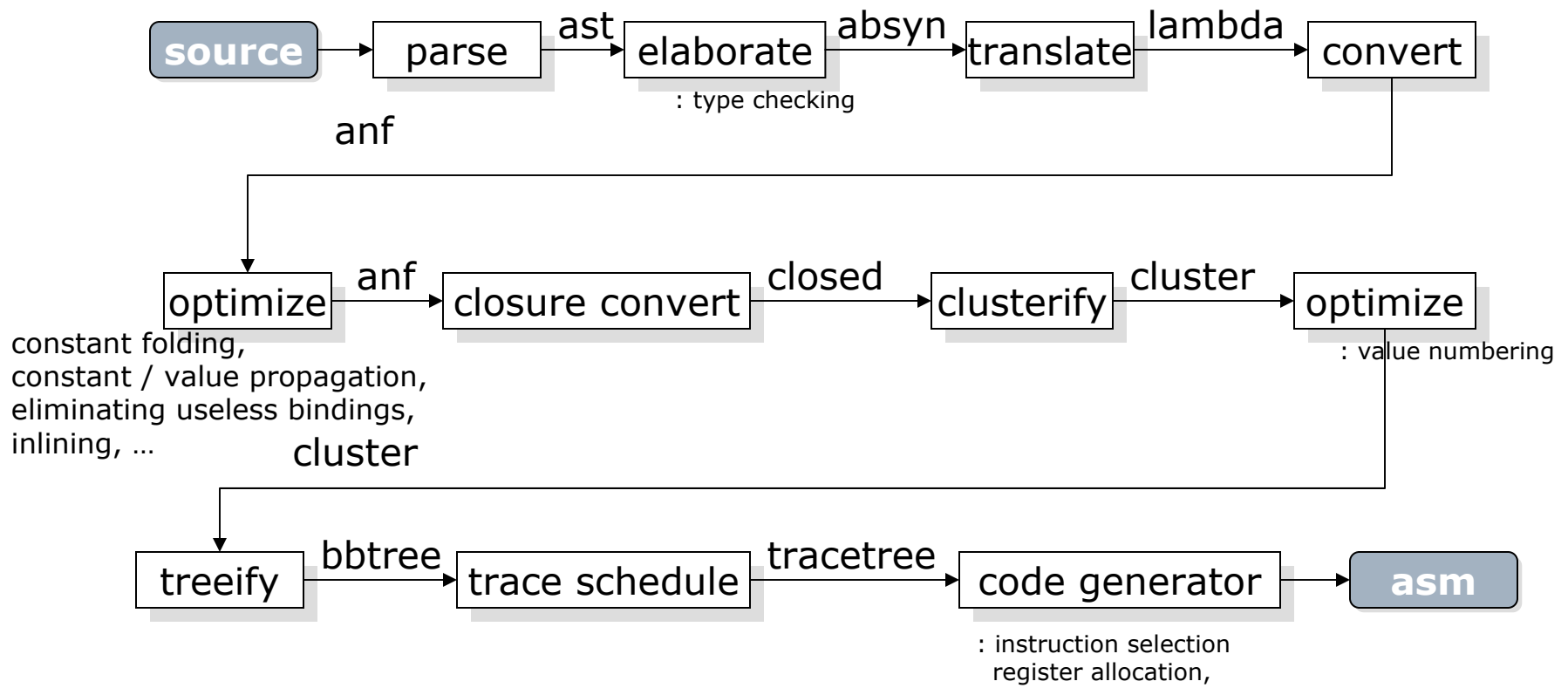
```
import sp1 as sp;  
import printf;  
  
main() {  
  bits32 s, p;  
  ...  
  s, p = sp (10);  
  foreign "C" printf (fmt, s, p);  
  ...  
}
```

(a) main.c--

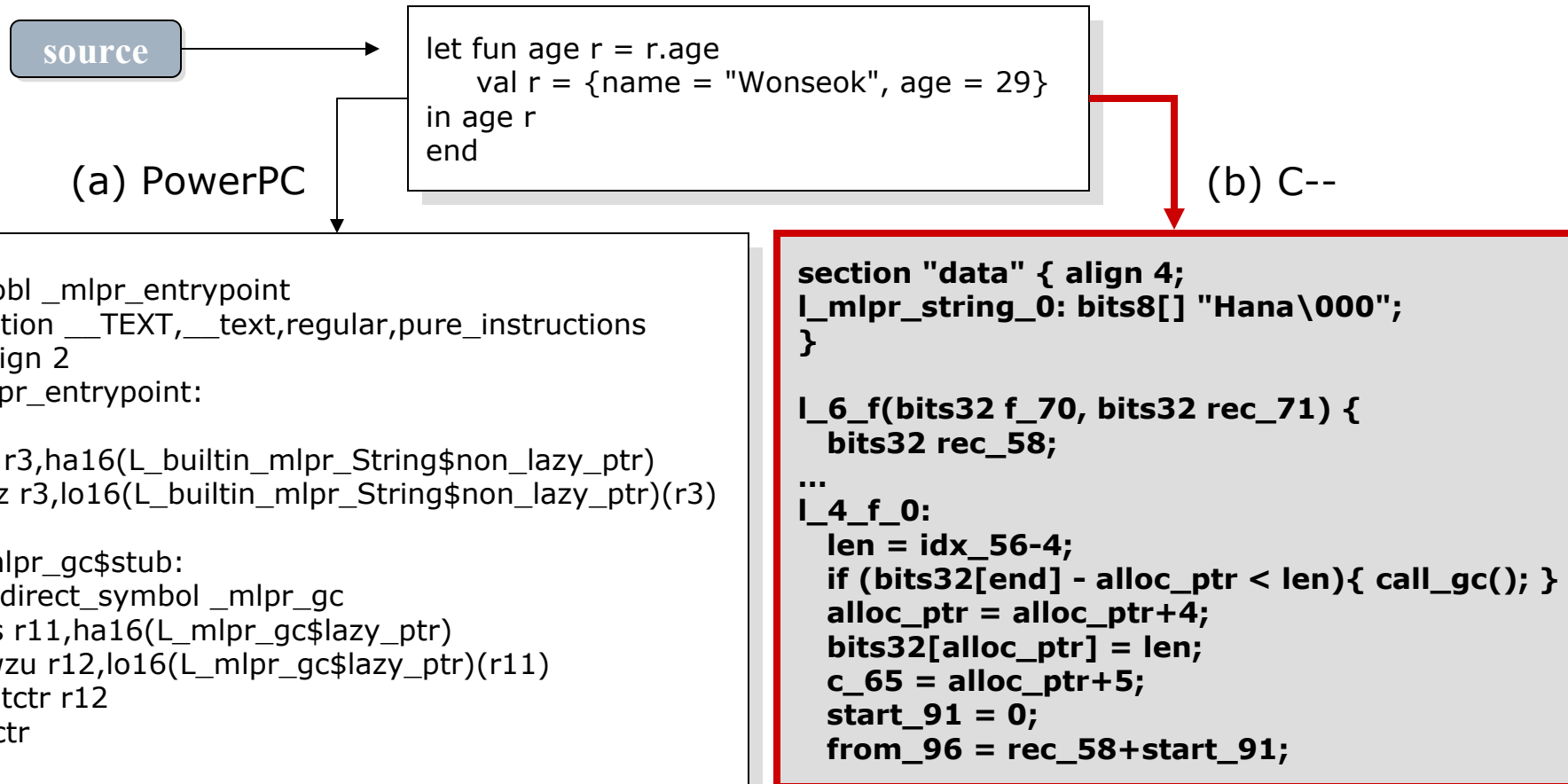
Case Study

- The MLPolyR frontend compiler for C--
 - MLPolyR is a tiny ML-like language with
 - Row polymorphism
 - Polymorphic record selection and sums
 - Functional record update
 - Extensible first-class cases
 - Mutable record fields
 - Type inference system with principle types
 - Type system based exception handling

The MLPolyR compiler

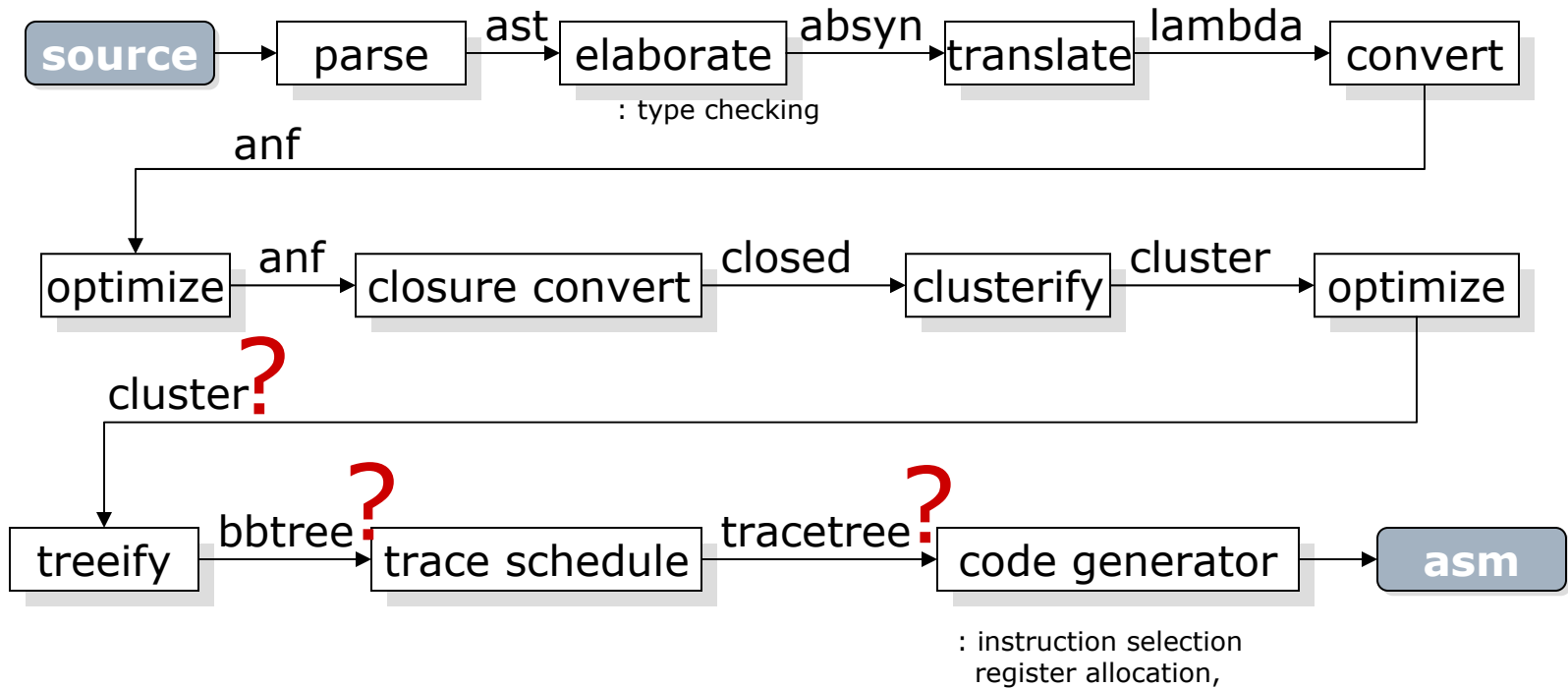


Goal: emit C-- w/ minimal efforts

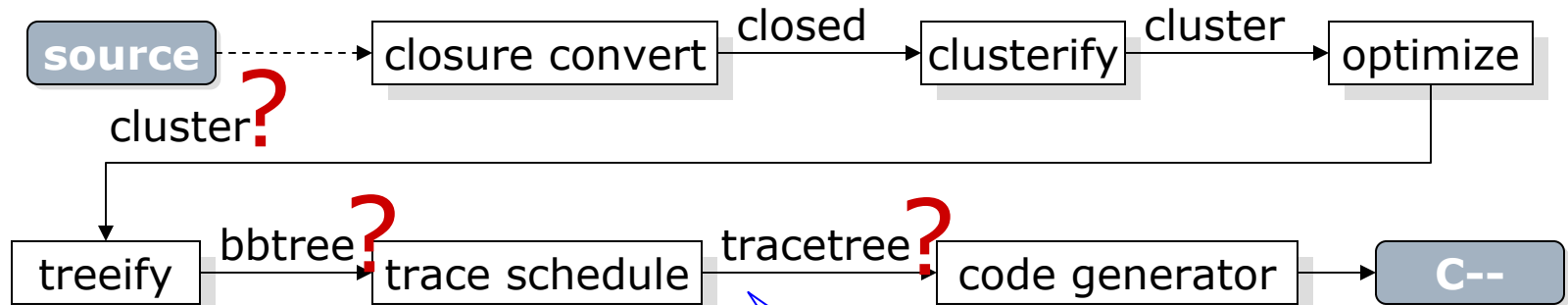


Design detail (1)

□ Which IL should be translated?



Design detail (2) BBtree or TraceTree?



```

I_1_fact(n_45):
  n_43 <- n_45
  goto I_0_fact
  _mlpr_main():
    n_43 <- 10
    goto I_0_fact

...
I_3:
  tmp_41 <- call I_1_fact ((n_39 - 2))
  return (n_39 * tmp_41)
I_0_fact:
  if n_43 > 0 then I_5 else I_4
  
```

```

I_1_fact(n_45):
  n_43 <- n_45

I_0_fact:
  if n_43 > 0
  then I_5
  else

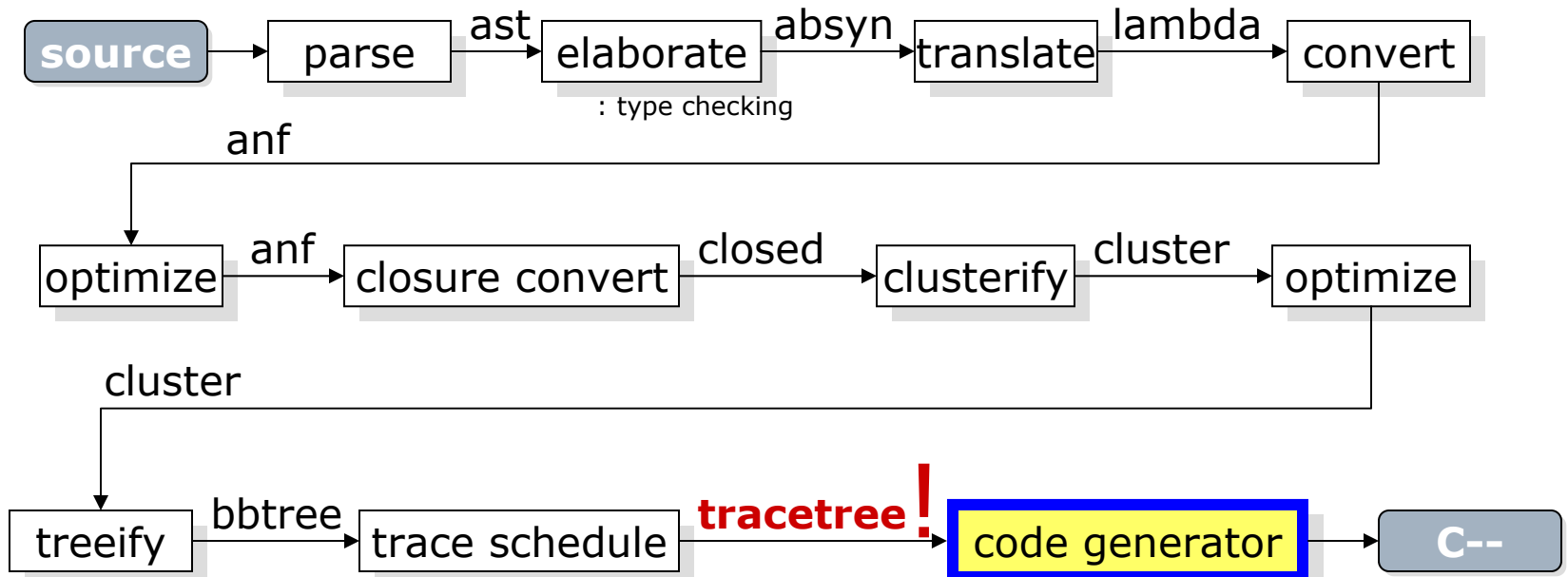
  I_4:
    return 2
  _mlpr_main():
    n_43 <- 10
    goto I_0_fact
  ...
  
```

```

I_1_fact(bits32 n_45) {
  bits32 n_43;
  ...
  I_0_fact_0:
    if (n_43>0) {goto I_5_0;}
  I_4_0:
    return(2);
  ...
}
mlpr_main() {
  bits32 n_43;
  ...
  return(2);
}
  
```

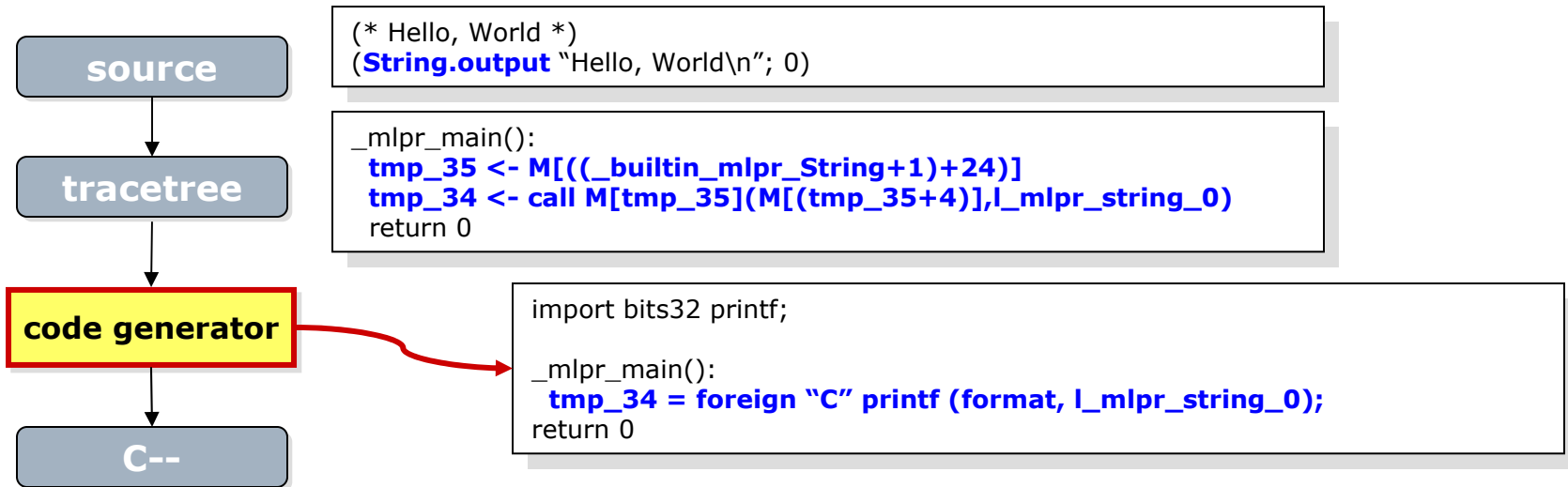
Design detail (3)

- Modification should be minimized.



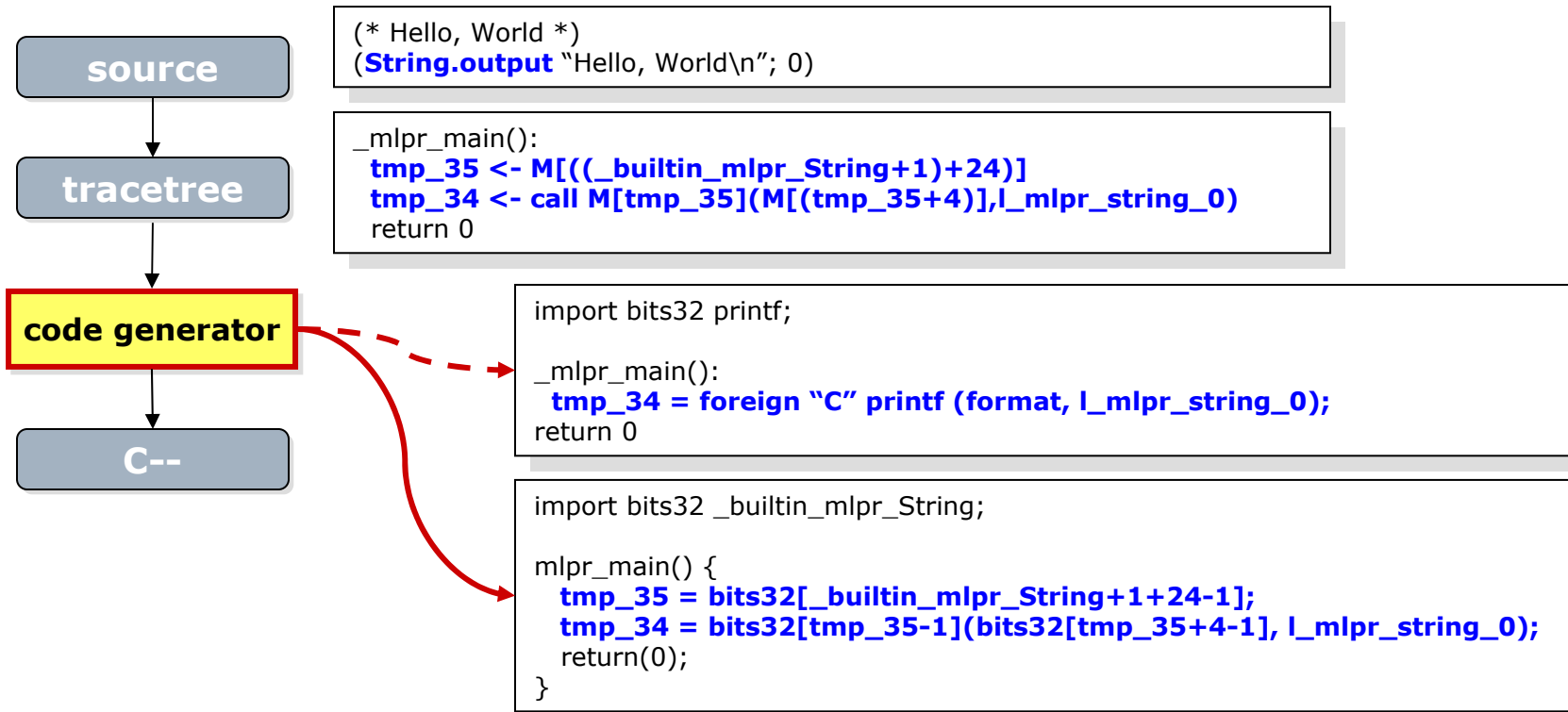
Design detail (4)

- A code generator anticipates changes.



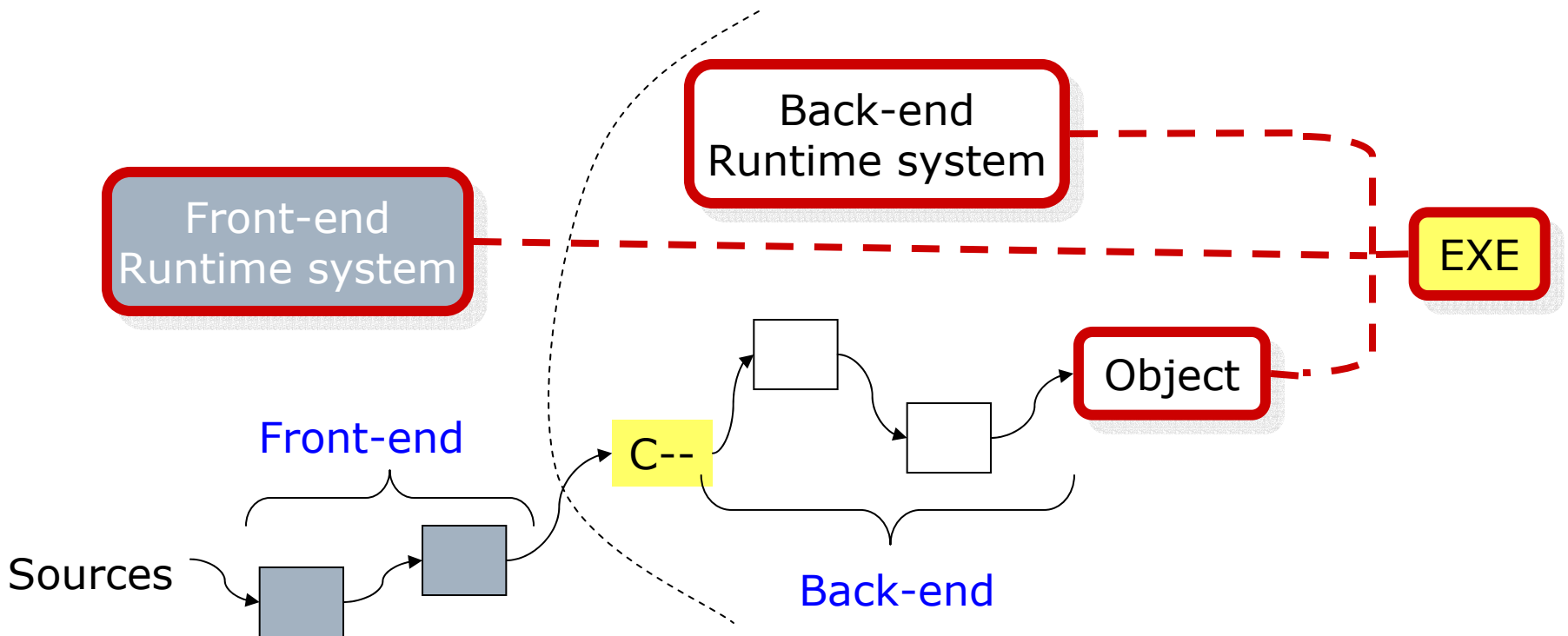
Design detail (5)

- A code generator anticipates changes.



Design detail (6)

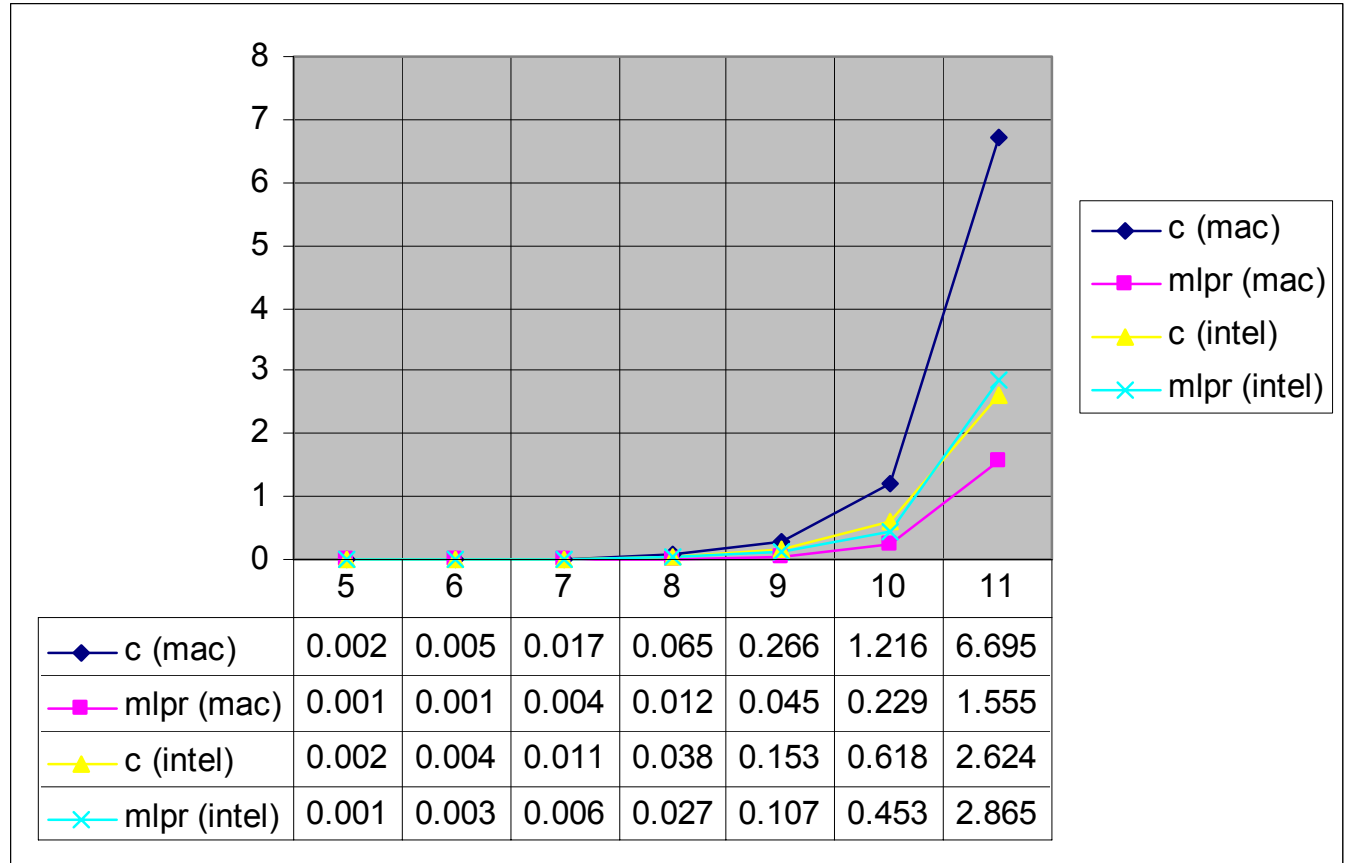
- GC uses C-- runtime service interface.



Benchmark: Ackermann function

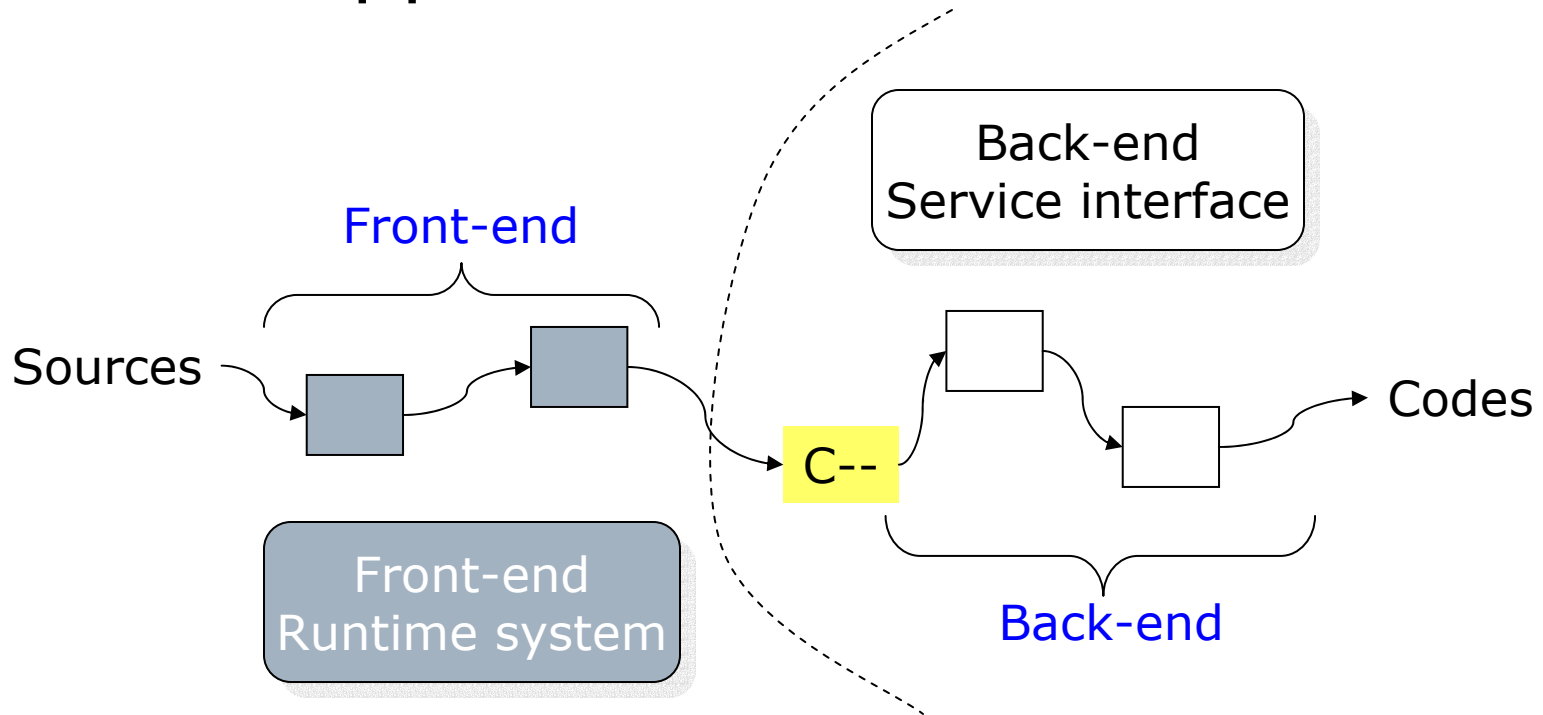
(* ackermann function *)

```
fun ack (m, n) =  
  if m == 0 then n + 1  
  (* else if m == 1 then n + 2 *)  
  else if n == 0 then ack (m - 1, 1)  
  else ack (m - 1, ack (m, n - 1))
```



Summary

- C-- is a portable assembler language that supports run-time services.



Reference

- The official website: www.cminusminus.org
- Simon P. Jones, Norman Ramsey, and Fermin Reig, "*C--: a portable assembly language that supports garbage collection*", PPDP, 1999
- Wonseok Chae, "*The MLPolyR front-end compiler for C--*", TTI-C, 2007
- Paul Govereau, "*Tiger Example front-end compiler for C--*", PLDI, 2004