Assembler

Ettore Speziale

Politecnico di Milano

Contents

Contents

The Next Step to Executable Code

ACSE emits assembly code:

- low level language (for humans)
- too much verbose (for CPUs)

The assembler translates assembly code to machine code:

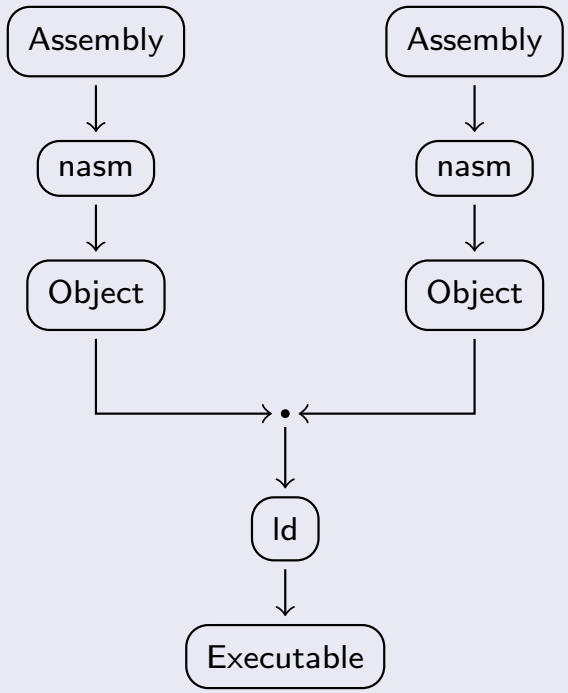
- can be directly executed by CPUs

To get final executable you need a linker.

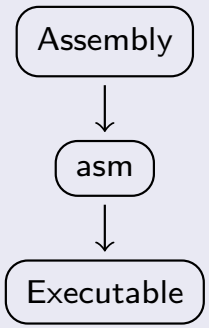
Comparing Assembling Tool-chains

ACSE is very simple:

GNU



ACSE



Linker not needed:

- only local symbols
- no functions
- only one module

Contents

The ACSE ASseMbler

A very simple assembler:

- parse an assembly file
- generates object code for the MACE machine

It can be used as starting point to:

- implements simple optimizations
- implements compiler backends

The Assembly Language

Assembly is much more simple than LANCE:

- a list of segments

Two kind of segments:

data contains variables

text contains program

Hello world

```
.text
  ADDI  R1  R0  #72
  WRITE  R1  0
  ...
  HALT
```

- no variables
- no data section

Variables

Each LANCE variable gets an entry in the data segment:

LANCE

```
int a = 7;
int b[10];

write(a);
```

.WORD [a] a 4 byte variable

.SPACE [b] an array of 10 elements

[a] Init value = 7

[b] Zero initialized

Assembly

```
.data
L0 : .WORD 7
L1 : .SPACE 40
.text
     LOAD R1 L0
     WRITE R1 0
     STORE R1 L0
     HALT
```

Internal Representation I

All inside `t_translation_infos`:

Assembler state

```
typedef struct {
  t_list *code;
  t_list *labels;
  int codesize;
} t_translation_infos;
```

- data and code in the same list
- labels in a separate list

Global variable `infos` references the assembler state.

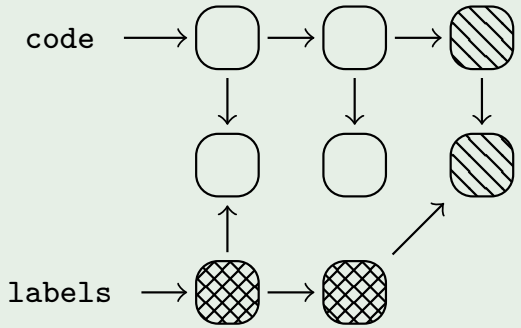
Internal Representation II

Internal state is trivial:

Source

```
.data
L0  :  .WORD  7
.text
L1  :  LOAD  R1  L0
       HALT
```

Assembler State by Picture



- instruction/data storage decoupled from CFG storage
- you can change execution order without changing instructions

Contents

Complexity

Compilers are not complex:

- simple algorithms because semantic must be maintained
- other tools (e.g. garbage collectors) much more complex

So, why many people do not think that?

- compilers are huge

So the problem is not the compiler, is the huge code-base:

- follow religiously coding conventions
- first read, then think, at last code

Contents

Bibliography

A. Di Biagio and G. Agosta.
Advanced Compiler System for Education.
http://compilergroup.elet.polimi.it, 2008.

Formal Languages and Compilers Group.
Software Compilers.
http://compilergroup.elet.polimi.it, 2010.