

# HrwCC

A self-compiling C-compiler.

Stefan Huber    Christian Rathgeb    Stefan Walkner

Universität Salzburg – VP Compiler Construction

June 26, 2007

# Overview

- 1 Introduction
  - Basic properties
  - Features
- 2 Selected details
  - Translation process
  - Compilation sample
- 3 Live demos
  - Features source file
  - Self-compilation

# Basic properties

## HrwCC

HrwCC is a **self-compiling** compiler for a **subset of C** whose output is

- linked with `hrwcc` and executed in the **VM `hrwvm`** or
- assembled with GNU assembler and executed natively.

# Features I

Some features of hrwcc:

- Single-pass compiler
- Strong featured preprocessor (i.e. `#define`, `#ifdef`, macros)  
⇒ Use abstract list implemented with macros.
- Separate compiling (using Makefiles, hrwcc supports `-g`, `-c`, `-O`, `-D`, `-E`)
- Simple optimizations available (`-O`)  
⇒ output of hrwcc itself reduced from 103.000 to 81.000 loc
- Calling libc-functions like `malloc`, `open`, `read`, `printf`

# Features II

Some selected features of the language:

- Arrays of primitive types and struct-instances
- Full pointer support  
Assign arrays to pointers, incrementing pointers, ...
- C-structs with assigning/passing struct-instances
- Lazy-evaluation of logical 'and' and 'or'
- Type safe assignments

# Translation process I

HrwCC creates GNU assembler code as output. Two possibilities:

- Linking with hrwcc and executing with own VM hrwvm
- Assembling and linking with gcc and executing under, lets say, Linux.

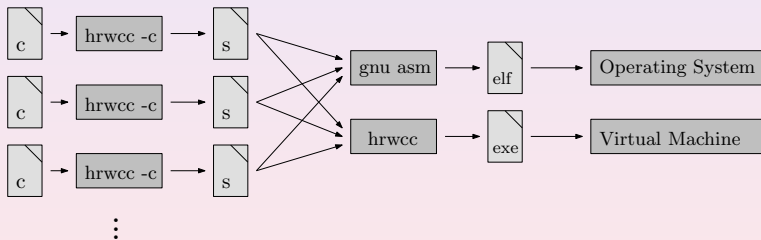


Figure: The build process with hrwcc.

# Translation process II

GNU assembler output has several advantages:

- Strong output language
- Strict separation of compiler and VM
- Very good testing capabilities
- Fast output when assembling with gcc
- Solving the library-function problem: Heap-memory, printf, etc.

⇒ Compiler was self-compiling at 28. April!

# A short compilation sample

In the following a short example will be compiled and the output will be shown.

```
#include "../..//include/hrwcccomp.h"

int main(int argc, char** argv)
{
    printf("%d arguments:\n", argc);
    return 2;
}
```



# The compiler output

```
.section .text
    #debug: func-def: ( int argc , char * * argv )
.globl main
.type main, @function
main:
    pushl   %ebp
    movl   %esp, %ebp
    subl   $0, %esp

    #debug: func-call: printf ( "%d arguments:\n" , argc )
    subl   $8, %esp
    movl   $syntab+0, 0(%esp)
    movl   %ebp, %eax
    addl   $8, %eax
    movl   (%eax), %eax
    movl   %eax, 4(%esp)
    call   printf
    addl   $8, %esp
    #debug: return: return 2 ;
    movl   $2, %eax
    jmp    main_ret

main_ret:
    movl   %ebp, %esp
    popl   %ebp
    ret

# The symbol table
.section .data
syntab:
    .string "%d arguments:\n"
```

# The optimized compiler output

```
.section    .text

.globl    main
.type    main,@function
main:
    pushl   %ebp
    movl   %esp,%ebp
    subl   $0,%esp

    subl   $8,%esp
    movl   $syntab+0,0(%esp)
    movl   8(%ebp),%eax
    movl   %eax,4(%esp)
    call   printf
    addl   $8,%esp

    movl   $2,%eax
    jmp    main_ret
main_ret:
    movl   %ebp,%esp
    popl   %ebp
    ret

.section    .data
syntab:
.string    "%d arguments:\n"
```

# The linker output

```
.section .text

call 3
pushl %eax
call exit

# .globl main [@3]
#.type main,@function
# main :  [@3]
pushl %ebp
movl %esp,%ebp
subl $0,%esp

subl $8,%esp
movl $17,0(%esp)
movl 8(%ebp),%eax
movl %eax,4(%esp)
call printf
addl $8,%esp

movl $2,%eax
jmp 14
# main_ret :  [@14]
movl %ebp,%esp
popl %ebp
ret

.section .data
# symtab :  [@17]
.string "%d arguments:\n"
```

# Features source file

Live: The features.cpp file

# Self-compilation

Live: `make -f Makefile.hrwcc`

# Bibliography I



Stefan Huber, Christian Rathgeb, Stefan Walkner.

Hrwcc website.

[http://www.cs.uni-salzburg.at/~ck/wiki/index.php?  
n=CC-Summer-2007.HrwCC](http://www.cs.uni-salzburg.at/~ck/wiki/index.php?n=CC-Summer-2007.HrwCC), 2007.

Thanks...  
...for your attention.