The Racket Virtual Machine

as an application of CS 4400
Some Racket Applications

- Hacker News
- Game Content
- Practical Typography
- Arc
- DC
- Pollen
- Racket
- Telescope Controller
- Synthesized Program
- Homework
- DrRacket
- Rosette
- Beginner Student
- Racket
- Racket
- Racket
Virtual Machines

- Racket Libraries & Programs
- Racket Virtual Machine
- C
- Operating System
- Memory Hierarchy
- Instruction Set Architecture
- Hardware
Virtual Machines

... 
Racket Libraries & Programs
Racket Virtual Machine
C
Operating System
Memory Hierarchy
Instruction Set Architecture
Hardware
Virtual Machines

- Racket Libraries & Programs
- Racket Virtual Machine
- C
- Operating System
- Memory Hierarchy
- Instruction Set Architecture
- Hardware
Virtual Machines

...  
Racket Libraries & Programs
Racket Virtual Machine
C
Operating System
Memory Hierarchy
Instruction Set Architecture
Hardware
Racket Virtual Machine

Racket Libraries & Programs

DrRacket  Slideshow

GUI  raco.exe  web server

Racket Virtual Machine

interpreter  primitives

JIT compiler  FFI  garbage collector
The Racket runtime system is implemented in C
Representing Numbers

- Representing fixnums: `SCHEME_INTP` and `SCHEME_TYPE` (scheme.h)
- ADD for fixnums (numarith.c)
- `SCHEME_RATIONAL_FROM_FLOAT` for `inexact->exact` (ratfloat.inc)
x86-64 Machine Model

Just-in-Time (JIT) compiler:

- ARITH_ADD

`jitarith.c`
Representing Control Flow

Just-in-Time (JIT) compiler:

- "unbox" implementation
  - `jitinline.c`

- `list_ref_code` implementation
  - `jitcommon.c`

See also `github.com/mflatt/jit-demo`
Representing Procedures

• unsafe/ffi

(define atoi
  (get-ffi-obj "atoi"
    #f
    (fun string -> int)))

• backtrace

• continuations
Arrays

• `array-ref`
Structures

- `Scheme_Object`  
  `scheme.h`

- `Scheme_Bignum`  
  `schpriv.h`

- `Scheme_Small_Bignum`  
  `schpriv.h`

- `Scheme_IR_Local`  
  `schpriv.h`
Optimization

- `scheme_application_type` case in `scheme_do_eval` 
  `eval.c`

- `XFORM_ASSERT_NO_CONVERSION` and:
  - `fd_write_string` vs.
  - `fd_write_string_string_slow` 
  `port.c`
More on Optimization

- Branch-prediction interaction in
  `scheme_generate_non_tail_call`  `jitcall.c`
Memory Hierarchy, Locality, Caches

- repair_heap's fused loops for the SIZE_CLASS_SMALL_PAGE case

```c
newgc.c
```
Linking

The \texttt{unsafe/ffi} functions work by dynamically loading shared libraries

- \texttt{ffi-lib} uses \texttt{dlopen}
- \texttt{get-ffi-obj} uses \texttt{dlsym}
ELF and Relocation

`raco exe` creates an executable by

- copying a stub binary that links to the Racket runtime system
- adding a new ELF section to hold bytecode for the Racket source

`collects/compiler/private/elf.rkt`
Processes

Racket runs `/bin/uname` to get the result of

```
(system-type 'machine)
```

`string.c`
More on Processes

The **subprocess** execs an arbitrary program

Implementation uses **fork** and **execve**, and **waitpid**

```c
port.c
place.c
```
File Descriptors

Racket’s I/O uses file descriptors directly

- `fd_get_string_slow`
Signals

• SIGINT handler in main

main.c

• SIGCHLD handler related to subprocess

port.c
Garbage collector allocates pages using \texttt{mmap}.

Write permission is disabled to implement a \texttt{write barrier} for generational collection.

Handler calls \texttt{designate\_modified\_gc}.
Dynamic Memory Allocation

- allocate

newgc.c
More on Memory Allocation

- `do_malloc` uses a free list
- Segmented allocation
Garbage Collection

• Bootstrap with conservative collector
• Convert C code to cooperate with precise GC
• Production GC is fairly complex

See also github.com/mflatt/jit-demo
Network Programming

• DNS

net/dns.rkt
More Network Programming

- Web server
- `raco pkg`
- Git checkout `net/git-checkout.rkt`
Concurrency

• File and network reads are multiplexed internally

• `MZ_GETADDRINFO`, which calls `getaddrinfo` in a thread

network.c
Synchronization

• Mutex at Racket-thread level protects hash tables (e.g., `hash_table_count`)  

• GC keeps a list of threads for cooperation on macOS  

list.c
...And More

What topics crucial to Racket weren't covered in CS 4400?

- Programming and data structures
- Interpreters and compilers
- Databases
- GUIs and graphics
- Rules and strategies for portability