Runtime Library Notes

Directory Structure

```
sd8516/                  project root
  runtime/
    crt0.sda             startup stub -- CALL main
    librt/               runtime functions
      divhi3.c
      modhi3.c
      divsi3.c
      muldi3.c
      ...
      softfloat/         the SF/DF families
    include/             freestanding headers (mine & what clang supplies)
    tests/               tests with checks
```

Prep Work

- Helper scripts were created this time: `c2sda`, `c2ll`, `ll2sda` and `runtests`.
- Next I had to modify VC-4 to take .sda files at the command line (to speed the dev cycle).
- Next I created the above directory.
- I will have to modify vc4 to look in ~/.vc4 if the files it needs are not in the pwd.

Fleshing it out

Many standard implementations reveal deficiencies in instrinfo and iseltargetlowering.

Test Suite

I collected all the tests into one file.

Roadmap

- putchar.sda (the only assembly only file so far – similar to tinyc v1)
- print_str
- print_dec
- lib1 (stuff for missing instructions)
- libctype
- libstr
- libstdlib

then

- stdint.h, limits.h, stddef.h, stdbool.h
- stdalign/stdnoreturn/iso646
- finish string.h (memchr, strncat, strpbrk, strspn, strcspn, strtok)
- safe-string extras (strlcpy/strlcat, strcasecmp/strncasecmp, memmem),
- div/ldiv, rand/srand, qsort/bsearch, assert.h
- official puts/getchar.

Roadmap which requires additional development:

- malloc/free/calloc/realloc
- then strdup/strndup

Phase III

- stdarg.h for varargs and formatted output. very high value target (enables printf)
- printf/sprintf/snprintf/vsnprintf and sscanf.

Later:

- compiler-facing primitives LEA, MOVSX, CMOV, get CASETAB to emit.
- 64-bit integers. muldi3, udivdi3/umoddi3, divdi3/moddi3.

Floating Point Roadmap:

- Start writing math.h and see what happens.
- strtod/atof, %f for printf.

File IO Roadmap:

- fopen/fread/fwrite/fseek/fclose and FILE* streams wrapping *(see INT 15h/OPFS/FCB)

Other:

- time.h, calendar math, binding time() to the hardware clock
- Graphics and Sound libraries
- access to PPU and ASU

Arena Allocator

“A growable, chunked, monotonic region allocator.”

This is very similar to the NVAR system I wrote for BASIC. Hanson uses “arena” in C Interfaces and Implementations; the academic literature says “region”. “Bump allocator” is the name of the mechanism, advancing a pointer within a chunk, but the whole structure, with chunked growth and free-everything-at-once teardown, is the arena.