The WIND Assembler and Interpreter Guide
Duane A. Bailey
November 15, 2011 (software version 2.135)

The WIND Assembler

The WIND assembler and interpreter (accessible with the commands `wia` and `wii`) are very similar to their WARM counterparts. This guide documents the few minor differences and highlights the optional features that, while available in both WARM and WIND systems, are typically used in the WARMVirtual Emulator (WAVE) project.

General Instruction Syntax

Students of the x86 instruction set will generally be familiar with the WIND assembly syntax, but there are a few important syntax-related observations.

- As with the AT&T style syntax of the x86 instruction set, all two-operand WIND instructions are written in the following form:

  ```
  label: opcode  source, destination     ; comment
  ```

  Typically the source and destination operands contribute values to the computation, and the result is written to the destination. This is in contrast the the WARM, which has a 3-operand syntax with the destination on the left.

- When specifying immediate values, the value is introduced with the dollar sign (`$`). (In the WARM, an `#` is used.)

- Unlike the x86 assembler, the WIND assembler does not require leading percent (`%`) characters on register names. In a similar stab at simplification, there are no traditional register name aliases (`eax`, etc.), though the programmer could introduce those through register equates.

- The syntax of particular addressing modes, of course, differs between the WARM and WIND assemblers. The WIND formats are described in detail in the *WIND Architecture Guide*.

- The software trap values (documented in the *WARM Assembler and Interpreter Guide*) are parallel in both architectures. These values should generally be accessed by the pre-defined symbols rather than by value. The particular values may change over time. The symbols will not.

Assembler Directives

All assembler directives are paralleled between the WIND and WARM assemblers. Please refer to the *WARM Assembler and Interpreter Guide* for details.

It is important to note that the WIND instructions occupy 1, 2, or 3 words of memory depending on the number of displacements or immediate values are involved in the instruction. For this reason, programmers may require an occasional `.align 4` to generate predictable target addresses for computed branches.

Running the WIND Assembler

Similar in functionality to the WARMassembler, the WIND assembler is invoked with the `wia` command. Its simplest form is

```
wia -l program.s
```

This command assembles the instructions in `program.s` and writes a machine code object file in `program.o`. This file contains all the information needed to interpret the program. When the `-l` switch is thrown, a listing file, `program.lst` is found in the current directory.

The WIND Interpreter

The WIND interpreter, called `wii`, parallels the WARM interpreter in nearly every way.

Of particular interest to the WAVE designer is the load extra overlay feature, made available with the `-x` switch:

```
wii -x warm-file.o wind-file.o
```

This command causes the file `wind-file.o` to be loaded (including symbols) into the store. The file `warm-file.o` is an object file targeted for the WARM. This file is mapped to memory on the invocation of the software trap, `SysOverlay`:

```
lea    loadArea,r0
trap   $SysOverlay
```

In this code, an address, `loadArea`, is stored in `r0`. The `SysOverlay` trap causes the file `warm-file.o` to be read into memory starting at the location indicated by the address, `r0`. When the overlay is loaded, the symbols associated with the overlay are ignored. Any instructions found in the overlay file, of course, are not understood by the WIND interpreter. That is, of course, the purpose of the WAVE project. (Overlays may, of course, be object files targeted for the same architecture. This mechanism might be used for a primitive form of dynamic loading.)