WASM

Assembler/Disassembler
For Wang Programmable Calculators
November 26, 2023

Table of Contents

INTRODUCTION

‘wasm’ is an Assembler and Disassembler for Wang 600/700 Programmable Calculators. It recognizes, but does not require, certain file type suffixes. Any suffix of the form “.w6x” (where ‘x’ is any letter) indicates Wang 600 code. Similarly, any suffix like “.w7x” indicates Wang 700 code. A suffix with the last letter ‘t’ is recognized as a tape image, which has additional codes to handle the end of the image.

ASSEMBLER

The assembler converts opcodes (and operands where appropriate) into instruction codes. Comments begin with a semicolon character and extend for the rest of the current line.

The special pseudo-op “ENTER” is used to make entering numbers easier. The operand for “ENTER” is a string of decimal digits with optional ‘E’, ‘-’, and ‘.’ characters. This string represents the sequence of keystrokes to enter the number, and NOT a standard number. In many cases the string representation will be the same as the number, but it need not be. For example, “1.1-”, “-1.1”, “1.-1” all represent the number -1.1. Additionally, the operand may be “**&***symbol*” to reference a symbolic register number (see **.REG** and **.SREG**), in which case a 3-digit entry sequence is coded for the register number.

Errors in assembly are noted by a letter in the first column of the listing output. The lines containing errors are also sent to stderr in the case that no listing is being printed to stdout. The error letters and their meanings are in the “ASSEMBLER ERRORS” section. The assembler uses two passes in order to resolve symbolic labels, but note that syntax errors may result in step numbers that are different from successful assembly.

DISASSEMBLER

The disassembler may be used to convert program or tape images into source code, allowing existing programs saved from a Wang Programmable Calculator Simulator to be turned into source code, or simply viewed (listed) in a symbolic form.

The disassembler looks for sequences of number entry commands and combines those into a single “ENTER” pseudo-op. It also converts ALPHA text sequences into more-natural strings.

COMMAND LINE

The following command line options are used to control the behavior of ‘wasm’:

600
: Use Wang 600 instruction codes and mnemonics. Also allows 601, 602, 611, 612, 607, or 606 to specify the default output device.

700
: Use Wang 700 instruction codes and mnemonics. Also allows 701, 702, 711, 712, 707, or 706 to specify the default output device.

docs
: Do not assemble/disassemble, only print a list of codes and mnemonics for the specified machine type.

tape
: Treat image file (input or output) as a cassette tape image, adding extra codes to mark the end of the image. Each End Program (END) code will appears as two successive END codes, and the end of a data image will have END followed by 15-15.

rom
: Treat image file (input or output) as a Wang 600 Expansion ROM image, padding with STOP codes to fill the remainder of the ROM. This also affects the assignment of subroutines to symbols.

asm=*file*
: Assemble *file* using the (implied or specified) machine type mnemonics and syntax. By default, a listing is produced on stdout and the object code is written to a file named “a.out”.

nolst
: Do not produce any assembly listing.

list=*file*
: Produce an assembly listing written to *file*, instead of sending to stdout.

out=*file*

Write assembled object code (or tape image) to *file*, instead of “a.out”. Or write disassembly output to *file* instead of stdout.

path=*dir...*

Use the list *dir…* to search for **.INCLUDE** files if they are not found relative to the current directory. Directories may be colon or semicolon separated, and will be searched in the order specified.

dat=*file*

Assemble *file* into data using the (implied or specified) machine type. By default, output is written to a file named “a.out”. *file* contains a list (one per line) of numbers. Prefixes the data with one register block containing the count of data items. May be used with **raw** (only data, no count or END) or **tape** (add extra END byte for tape format) to alter output format.

dis=*file*

Disassemble the program image *file* and send to stdout. By default, this produces a listing-style output (containing step numbers and object code).

raw

The disassembly output is source code, not listing. This code should be ready for re-assembly.

Note that it does not make sense to specify both “dis=” and “asm=” in the same command. If both are specified, “asm=” will be ignored.

OPERAND FORMATS

ALPHA

The “ALPHA” mnemonic, which represents the “α” key on the Wang 600 or the “WRITE ALPHA” key on the Wang 700, takes an operand that may be a single specific key (mnemonic) or a numeric code (e.g. 00-00), or a string enclosed in double quotes. In the case of a string, the ALPHA command is coded as a variable-length command that is automatically terminated by the appropriate code based on machine type. Within the string, shift codes are handle automatically. The following special codes (escapes) are used to represent special characters:

“\r” RETURN-INDEX (carriage-return line-feed)
“\b” BACKSPACE
“\t” TAB
“\i” INDEX (line-feed)
“\v” REVERSE INDEX
“\h” “½” (OutputWriters)
“\h” Move pen to home position (flatbed plotter)
“\q” “¼” (OutputWriters)
“\c” “¢” (OutputWriters)

“\n” DC2 a.k.a. PUNCH-ON (Teletype)
“\f” DC4 a.k.a. PUNCH-OFF (Teletype)
“\p” Draw line to new position (flatbed plotter)
“\m” Move pen to new position (flatbed plotter)
“\z” Set character size (flatbed plotter)
“\s” Set character spacing (flatbed plotter)

Not all output devices support all these characters.

Prefixing a printable character with “|” encodes the plot variation of the character, for use with plotting devices (Plotting OutputWriter or flatbed plotter).

REGISTER DIRECT

Register direct instructions use the decimal value of the register index, the same number that would be used for indirect register access (not the coded value placed in the program step). For the Wang 700, this includes automatic handling of the “register + 100” cases. In conjunction with **.REG** and **.SREG** symbolic register names, the notation **&***name* may be used in place of a decimal value.

PRINT / WRITE

The PRINT (Wang 600) or WRITE (Wang 700) instruction is followed by a code that indicates the format used to print the number. The Wang 600 uses a format that specifies a letter “tag” and the number of decimal places separated by a slash, for example “X/02”. The Wang 700 uses a format that specifies the number of places to blank-pad in the whole-number portion and the number of decimal places, represented as a standard program code, for example “05-02”.

MARK / SEARCH

Instructions that use a statement label may specify the label as either a standard code (“00-00”) or by the key mnemonic (“E0”). The disassembler favors the key mnemonic. In addition, a symbolic label may be used, of the form “**&***string*”. One-step subroutines may be referenced symbolically using “**$***string*”. Symbolic labels are assigned values after the first pass, and will exclude any labels explicitly defined, as well as the END PROG code. In the case of the Wang 600, symbolic subroutines will use either the normal program range or the ROM range, depending on the context of the assembly.

ASSEMBLER ERRORS

The following error letters are used:

V Invalid characters in ENTER operand.
O Invalid opcode mnemonic.
S Syntax error (missing operand).
P Operand error.
R Register value error.
F Printer format error.

I I/O operand error.
X Register data defined in ROM or .PROG not first.
U Undefined symbolic label/register.
M Multiple definitions of symbolic label.
Z Overflow of program memory.

ASSEMBLER PSEUDO-OPS

The following pseudo-ops are recognized:

.PROG *start*[*end*[*regs*]]

Define program space to be used. The starting program step is specified by *start*. The last program step is specified by *end* (default to memory size of machine). The starting step for registers outside the program is specified by *regs*. The default if no *regs* is specified is to use the space after the last program code. This directive is mainly used for program overlays and to specify where registers are allocated. For most programs that do not use embedded registers, the actual program step numbers don't matter.

.REG [*symbol*] ["*string*" | *value*]

Define register space inside the program. The program is padded with STOP instructions to the next register location. If *symbol* is specified, it will be assigned the register number corresponding to the location in the program. It may be used by prefixing with "&" in register-direct instructions. If the machine stores two registers in a program block (as the 700 does), then a second symbol and second value may be specified, separated by a comma. If *symbol* is "-" then no symbol is generated (but the value field is still recognized). The *string* is a hexadecimal string of 16 digits, representing the codes to be stored in each digit location in the register. The *value* is a floating point value in standard notation, for example "1" or "1.2e6".

.SREG *symbol*[**,***symbol*] [*count*]

Define uninitialized register space outside the program. The default *count* is one register block (two registers on the Wang 700). The *symbol* will be assigned the lowest register number of the space, and may be optionally followed (comma separated) by another *symbol* which is assigned the highest register number of the space. A symbol name "-" is used to assign no symbol (place holder for specifying *count*). This register space is not part of the saved program image, but will be assigned space immediately following the image (in the order that the **.SREG** directives appear). The start of this register space may be overridden using the **.PROG** directive.

.OUT *device*

Specify the output device type that applies to subsequent ALPHA text instructions. This determines the character set available and how strings are converted to Wang character codes. The device value must match a valid output device for the Wang machine family, for example the model 601 is the Output Writer for the Wang 600, while 701 is the Output Writer for the Wang 700. The following models are supported:

601/701	Basic Output Writer
602/702	Plotting Output Writer

611/711	Input/Output Writer
612/712	Flatbed Plotter
607/707	Teletype

.INCLUDE *file*

Include the specified file at the current location in the program. *File* may be absolute or relative to the current directory when running "wasm". In addition, the command argument **path=** may be used to specify a search path. Note that *file* is appended whole to each *dir* in the search path, so if *file* contains a directory component, that will also be used in the search.

.EXT *label...*

Declare label(s) as being defined outside of the program module being assembled. Labels are separated by blanks/tabs. Symbolic labels are not supported as external references, however symbols may be assign to external labels using the **.DEF** directive.

.EXTROM *label...*

Declare ROM label(s) as being defined outside of the program module being assembled. This is only valid for Wang 600 programs.

.DEF *symbol label*

Pre-define *symbol* to be value *label*. Use "**&***string*" or "**$***string*" notation for *symbol*. *label* is either an instruction opcode or numeric code value. This is mainly used for entry points that require a consistent user or program interface, for example the expansion ROMs on Wang 600. another use case would be for modular/overlay program sections. There must still be a "**MARK** *symbol*" somewhere in the program, or a **.EXT** directive for *label*.

In addition, a Wang 600 assembly may use the instruction mnemonics **@SEARCH** and **@CALL** to automatically produce the correct opcode for expansion ROM or program memory, depending on the **rom** command line option.

EXAMPLES

This example uses the **.REG** psuedo-op to create a register within a program that contains a pattern for a blank display on the Wang 600:

```
...
      RECALL &blank
      ALPHA STOP
      ALPHA STOP
...
      .REG blank "FFFFFFFFFFFF00A"
      END
```

There is currently no way to tell the disassembler about sections of a file that contain register data.

(end of document)