MIPS_ASM User Manual

1) INTRODUCTION

MIPS_ASM is a simple assembler. It does not assume any type of section. Data and instruction can be mixed anywhere.

2) SYNTAX

Basic instruction has the following format:

```
Label:  Instruction    operands              ; Comment
```

Label — must be immediately followed by colon. Label can be in front of an instruction or can be on a line by itself.

Comment — can be on any line after an instruction or on a line by itself.

Instruction — is in form described in Volume II: The MIPS32 Instruction Set.

The following are examples of instruction:

```
#define XXX    (bat_dau+4)

                       org     0x12300000
                       ;  test 1 program
data:
                       org     0x12310000
data1:
bat_dau:
                       lui     r0,@data
                       ori     r0,r0,data&0xffff             ; r0 = data:
                       lui     r1,@XXX
                       ori     r1,r1,XXX&0xffff
```

The following is the list file generated by MIPS_ASM:

```
                      1 #define    XXX                            (bat_dau+4)
                      2
                      3            org 0x12300000
                      4 ;  test 1 program
                      5 data:
                      6            org 0x12310000
                      7 data1:
                      8 bat_dau:
3C001230 12310000     9            lui      r0,@data
34000000 12310004    10            ori      r0,r0,data&0xffff    ; r0 = data:
3C011231 12310008    11            lui      r1,@XXX
34210004 1231000C    12            ori      r1,r1,XXX&0xffff
```

This example has label, instruction and comment examples.

This example also introduces ‘org’ statement. ‘org’ statement is used to set the address within the CPU address space. Address does not have to be in increasing order. MIPS_ASM will check for address conflict in pass 2.

This example also introduces the define statement in order to define a complex expression. The following are operators that can be used:

'+'	Add
'-'	Subtract
'*'	Multiply
'/'	Divide
'%'	Modulo
'&'	Bit wise and
'|'	Bit wise or
'^'	Bit wise xor
'~'	Bit wise not
'@'	Upper 16-bit value of a 32-bit value

That is it for instruction. For data, there are two statements: 'ds' and 'dw'.

'ds' is used to define the number of 32-bit locations. For example:

```
data:          ds      32
```

'dw' is used to define 32-bit values. For example:

value: dw 0, 1, 2, "phongle", 13, 10, 0

3) FILE GENERATED BY MIPS_ASM

MIPS_ASM expects a file with .asm to be the input file. For example, if the input file is sort.asm, the following command is used to compile sort.asm:

- mip_asm sort

mips_asm will generate the following files:

sort.111	Pass 1 listing of source file in pretty print format.
sort.lst	Pass 2 listing of source file in pretty print format.
sort.sym	file of labels and corresponding addresses. Symbols begin with '_' will not be stored in this file.
sort.bin	ASCII file of compiled code and data values.

These files are used by the MIPS simulator.