

# MIPS Assembly

---

Copyright (c) 2011-2014 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice and Octave.



and \$s3, \$s1, \$s2  
or \$s3, \$s1, \$s2  
xor \$s3, \$s1, \$s2  
nor \$s3, \$s1, \$s2

$\$s3 = \underline{\$s1} \text{ and } \underline{\$s2}$   
 $\$s3 = \underline{\$s1} \text{ or } \underline{\$s2}$   
 $\$s3 = \underline{\$s1} \text{ xor } \underline{\$s2}$   
 $\$s3 = \underline{\$s1} \text{ nor } \underline{\$s2}$

andi \$s3, \$s1, 0x12FA  
ori \$s3, \$s1, 0x12FA  
xori \$s3, \$s1, 0x12FA

$\$s3 = \underline{\$s1} \text{ and } \underline{0x12FA}$   
 $\$s3 = \underline{\$s1} \text{ or } \underline{0x12FA}$   
 $\$s3 = \underline{\$s1} \text{ xor } \underline{0x12FA}$   
 $\$s3 = \underline{\$s1} \text{ nor } \underline{0x12FA}$

```
sll $t0, $s1, 4  
srl $t0, $s1, 4  
sra $t0, $s1, 4
```

```
$t0 = $s1 << 4  
$t0 = $s1 >> 4  
$t0 = $s1 >>> 4
```

```
sllv $t0, $s1, $s2  
srlv $t0, $s1, $s2  
sra v $t0, $s1, $s2
```

```
$t0 = $s1 << ($s2)  
$t0 = $s1 >> ($s2)  
$t0 = $s1 >>> ($s2)
```

---

int a = 0x12fa;

# \$s0 = a  
addi \$s0, \$0, 0x12fa

int a = 0x12fa34fa;

# \$s0 = a  
lui \$s0, 0x12fa  
ori \$s0, \$s0, 0x34fa

#	\$s0 = a	
	addi \$s0, \$0, 4	\$s0 = \$0 + 4
	addi \$s1, \$0, 1	\$s1 = \$0 + 1
	sll \$s1, \$s1, 2	\$s1 = \$s1 << 2
	beq \$s0, \$s1, target	if (\$s0 == \$s1) goto target
	addi \$s1, \$s1, 1	\$s1 = \$s1 + 1
	sub \$s1, \$s1, \$s0	\$s1 = \$s1 - \$s0
target:		
	add \$s0, \$s1, \$s0	\$s0 = \$s1 + \$s0

{	beq \$s0, \$s1, target
	bne \$s0, \$s1, target
	j target
	jr \$t0

```
addi $s0, $0, 4
addi $s1, $0, 1
sll  $s1, $s1, 2
j    target
addi $s1, $s1, 1
sub  $s1, $s1, $s0
```

```
$s0 =  $0  +  4
$s1 =  $0  +  1
$s1 =  $s1 << 2
if ($s0 == $s1) goto target
$s1 =  $s1 +  1
$s1 =  $s1 -  $s0
```

target:

```
add  $s0, $s1, $s0
```

```
$s0 =  $s1  +  $s0
```

```
beq  $s0, $s1, target
bne  $s0, $s1, target
```



## References

- [1] <http://en.wikipedia.org/>
- [2] M. M. Mano, C. R. Kime, "Logic and Computer Design Fundamentals", 4<sup>th</sup> ed.
- [3] J. Stephenson, Understanding Metastability in FPGAs. Altera Corporation white paper. July 2009.