ARITHMETIC

MIPS Assembly

Arithmetic Instructions

MIPS Instruction Set

- Arithmetic (ADD, SUB, MULT, DIV)
  - Logic, and Shifting Instructions
- Conditional Branch Instructions
- Function Call Instructions
- Load and Store Instructions.

1. Arithmetic, Logic, and Shifting Instructions

4.4.1 Arithmetic Instructions

	Op	Operands	Description
○	`abs`	*des, src1*	*des* gets the absolute value of *src1*.
	`add(u)`	*des, src1, src2*	*des* gets *src1* + *src2*.
	`and`	*des, src1, src2*	*des* gets the bitwise and of *src1* and *src2*.
	`div(u)`	*src1, reg2*	Divide *src1* by *reg2*, leaving the quotient in register `lo` and the remainder in register `hi`.
○	`div(u)`	*des, src1, src2*	*des* gets *src1* / *src2*.
○	`mul`	*des, src1, src2*	*des* gets *src1* × *src2*.
○	`mulo`	*des, src1, src2*	*des* gets *src1* × *src2*, with overflow.
	`mult(u)`	*src1, reg2*	Multiply *src1* and *reg2*, leaving the low-order word in register `lo` and the high-order word in register `hi`.
○	`neg(u)`	*des, src1*	*des* gets the negative of *src1*.
	`nor`	*des, src1, src2*	*des* gets the bitwise logical **nor** of *src1* and *src2*.
○	`not`	*des, src1*	*des* gets the bitwise logical negation of *src1*.
	`or`	*des, src1, src2*	*des* gets the bitwise logical **or** of *src1* and *src2*.
○	`rem(u)`	*des, src1, src2*	*des* gets the remainder of dividing *src1* by *src2*.
○	`rol`	*des, src1, src2*	*des* gets the result of rotating left the contents of *src1* by *src2* bits.
○	`ror`	*des, src1, src2*	*des* gets the result of rotating right the contents of *src1* by *src2* bits.
	`sll`	*des, src1, src2*	*des* gets *src1* shifted left by *src2* bits.
	`sra`	*des, src1, src2*	Right shift arithmetic.
	`srl`	*des, src1, src2*	Right shift logical.
	`sub(u)`	*des, src1, src2*	*des* gets *src1* - *src2*.
	`xor`	*des, src1, src2*	*des* gets the bitwise exclusive **or** of *src1* and *src2*.

pseudo-instructions

`l`oading an **`i`**mmediate value

- **`li`** is a pseudo-instruction that loads an immediate value into a register
- Pseudo-instructions are only understood by the MIPS assembler but not by the CPU (MIPS)
- More MIPS pseudo-instructions:
- **`blt, bgt, ble, neg, not, bge, li, la, move`**

We already used the 3 pseudo-instructions: **`li, la, move`**

`li` (`l`oad-`i`mmediate) pseudo-instruction

- `li $t0, 25`
- It is equivalent to:

```
addi $t0, $0, 25
```

Book page. 313

`la` (`l`oad-`a`ddress) pseudo-instruction

- `la $t0, 0x74A12`
- It is equivalent to:
  - `lui $t0, 0x0007`
  - `ori $t0, $t0, 0x4A12`
- `lui` = `l`oad-`u`pper-`i`mmediate (loads the upper 16 bits and the lower 16 bits with zeros)
- `ori` = `or-i`mmediate

Book page. 313

`move` pseudo-instruction

- `move $a0, $t4`
- It is equivalent to:
- `add $a0, $t4, $0          # $a0 = $t4`

Addition and Subtraction

Example-1

Add … Sub

`sub $t3,$t0,$t1`: (`$t3 = $t0-$t1`)

```
      .text
      .globl main
main:
      li   $t0, 9
      li   $t1, 6
      li   $t2, 7
      sub  $t3, $t0, $t1
      add  $t4, $t3, $t2

      li   $v0, 10
      syscall
```

What is the function that is implemented ?. `$t4`=?

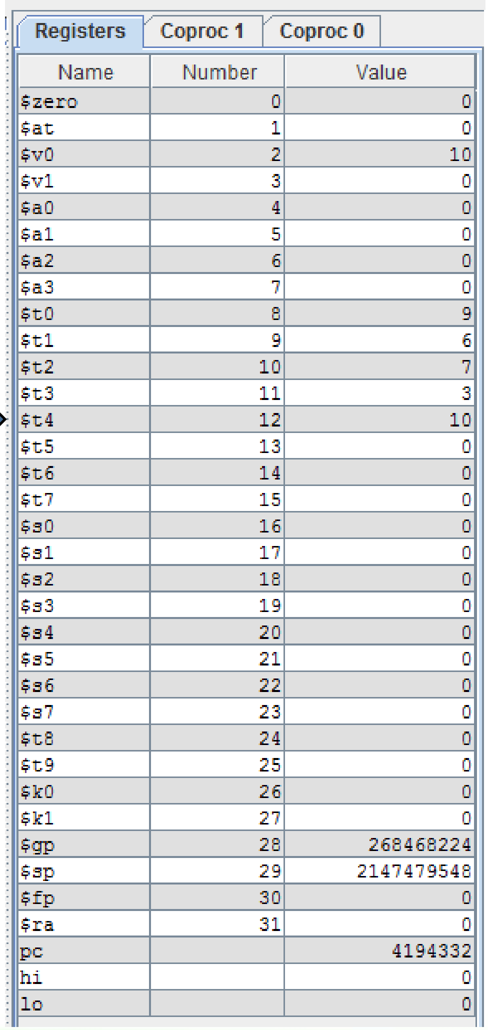
Add … Sub

`sub $t3,$t0,$t1`: (`$t3 = $t0-$t1`)

Function: ($t0-$t1)+$t2

```
      .text
      .globl main
main:
        li   $t0, 9
        li   $t1, 6
        li   $t2, 7
        sub   $t3, $t0, $t1
        add   $t4, $t3, $t2

        li    $v0, 10
        syscall
```


Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	9
$t1	9	6
$t2	10	7
$t3	11	3
$t4	12	10
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194332
hi		0
lo		0

Addition and Subtraction

Example-2

Trace the code

```
        .text
        .globl main
main:
        li      $t0, 9
        li      $t1, 6
        li      $t2, 7
        sub     $t3, $t0, $t1
        add     $t4, $t3, $t2
        la      $a0, msg            # prints message
        li      $v0, 4
        syscall
        move    $a0, $t4            # move to a0 for printout
        li      $v0, 1
        syscall
        li      $v0, 10             # exit
        syscall

        .data
 msg:   .asciiz  "The answer is:   "
```

Assemble … GO

Mars Messages | Run I/O

Clear

```
The answer is:  10
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	10
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	9
$t1	9	6
$t2	10	7
$t3	11	3
$t4	12	10
$t5	13	0

Multiplication

Example …

- Assume that our CPU is 3-bits. (| | |)
- Multiply: [111_2] x [111_2]
- The result is: [110001_2] …
- The [110001_2] can only fit in **Two** 3-bit registers

1	1	0
0	0	1

Two 3-bit registers

1	1	0	0	0	1
	hi register			**lo** register	

MDU

- MIPS has a special Multiplication/Division Unit (MDU)
- The MDU multiplies two 32-bit numbers and stores the result in 64-bits, which in hardware requirements is **two** **32-bit registers**)
- The **two** **32-bit registers** are named:
  - **`mfhi(hi)`**
  - **`mflo(lo)`**.

Multiplication/Division

MIPS: Multiply and Divide

```
Mult   $t1, $t2   # Multiply ($t1*$t2) to produce a 64-bit number (hi,lo)
Mfhi   $t3        # move result to special 32-bit  register hi ($t3)
Mflo   $t4        # move result to special 32-bit  register lo ($t4)
```

```
Div    $t1, $t2   # Div ($t1/$t2)
                  # to produce a 32 -bit lo [$t1 / $t2  integer quotient ]
                  # to produce a 32 -bit hi [$t1 % $t2  remainder]
Mfhi   $t3        # move result to special32-bit  register hi ($t3)
Mflo   $t4        # move result to special 32-bit register lo ($t4)
```

mfhi ("**m**ove **f**rom **hi**")

mflo ("**m**ove **f**rom **lo**")

lo and **hi** are registers in MIPS

`lo` and `hi` registers

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

`Multiplication`

```
mult $t1, $t2
```

Example-3

Multiplication-A

```
# Multiplication ... exampleFirst.asm

        .text
        .globl main
main:
        li      $t1, 50
        li      $t2, 2
        mult    $t1, $t2

        li      $v0, 10# exit
        syscall
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	50
$t2	10	2
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194324
hi		0
lo		100

Multiplication

```
mult $t1, $t2
```

Example-4

Multiplication-B

```
        .text
       .globl main
main:
        li      $t1, 50
        li      $t2, 2
        mult    $t1, $t2
        mflo    $t3
        mfhi    $t4

        li      $v0, 10
        syscall
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	50
$t2	10	2
$t3	11	100
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194332
hi		0
lo		100

Multiplication

```
mult $t1, $t2
```

Example-5

Trace the code

```
        .text
        .globl main
main:
        li          $t1, 50
        li          $t2, 2
        mult        $t1, $t2
        mflo        $t3
        la          $a0, msg    # prints message
        li          $v0, 4
        syscall

        move        $a0, $t3    # move to a0 for printout.
        li          $v0, 1
        syscall
        li          $v0, 10    # exit
        syscall

        .data
msg:    .asciiz     "The answer is:   "
```

Assemble … GO

```
The answer is:  100
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	100
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	50
$t2	10	2
$t3	11	100

Multiply in MIPS; hi-lo registers

- $2_{10} = 0010_2$
- $50_{10} = 110010_2$
- $2*50 = 100_{10} = 1100100_2$

$$\underbrace{00000000000000000000000000000000}_{\text{hi}} \quad \underbrace{000000000000000000000000\ \mathbf{01100100}}_{\text{lo}}$$

mfhi ("**m**ove **f**rom **hi**")
mflo ("**m**ove **f**rom **lo**")

`mul` … (pseudoinstruction)

```
mul $t3, $t1, $t2
```

Multiply `$t1` by `$t2` and store the result to `$t3`

It is equivalent with:

```
mult  $t1, $t2
mflo  $t3
```

Only when the result is 32-bits = 1 (32-bit) Register

Division

`Div $t1, $t2`

Example-6

Divide

```
        .text
        .globl main
main:
        li      $t0, 24
        li      $t1, 8
        div      $t0, $t1

        li      $v0,10
        syscall
```

Registers	Coproc 1	Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	24
$t1	9	8
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
[illegible]		[illegible]
hi		0
lo		3

Divide in MIPS; hi-lo registers

- $24_{10} = 11000_2$
- $8_{10} = 1001_2$
- $24/8_{10} = 0_{10} + 3_{10} = 0000\ 0011$

$$\underbrace{00000000000000000000000000000000}_{\text{hi (Remainder)}}\ \underbrace{00000000000000000000000000000011}_{\text{lo (Quotient)}}$$

mfhi ("**m**ove **f**rom **hi**")
mflo ("**m**ove **f**rom **lo**")

Division

```
Div $t0, $t1
```

Example-7

Divide

```
        .text
        .globl main
main:
        li      $t0, 26
        li      $t1, 8
        div     $t0, $t1

        li      $v0,10
        syscall
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	26
$t1	9	8
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194324
hi		2
lo		3

Divide in MIPS; hi-lo registers

- $26_{10} = 11010_2$
- $8_{10} = 1000_2$
- $26/8_{10} = 2_{10}+3_{10} = 0010 \quad 0011$

$$\underbrace{00000000000000000000000000000010}_{\text{hi (Remainder)}} \quad \underbrace{00000000000000000000000000000011}_{\text{lo (Quotient)}}$$

mfhi ("**m**ove **f**rom **hi**")
mflo ("**m**ove **f**rom **lo**")

Division

Div $t0, $t1

Example-8

Divide Example (with: mflo/mfhi)

```
        .text
        .globl main
main:
        li      $t0, 26
        li      $t1, 8
        div     $t0, $t1
        mflo    $t3
        mfhi    $t4

        li      $v0,10
        syscall
```

mflo = ?

Mfhi = ?

$t3 = ?

$t4 = ?

Assemble GO

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	26
$t1	9	8
$t2	10	0
$t3	11	3
$t4	12	2
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194332
hi		2
lo		3

mflo = 3

Mfhi = 2

$t3 = 3

$t4 = 2

26/8 = 3 and 2/8

Division

```
Div $t0, $t1
```

Example-9

Divide

```
        .text
        .globl main
main:
        li          $t0, 24
        li          $t1, 8
        div         $t0, $t1
        mflo        $t3
        mfhi        $t4
        la          $a0, msg            # prints message
        li          $v0, 4
        syscall

        move        $a0, $t3            # move to a0 for printout
        li          $v0, 1
        syscall
        li          $v0,10              # exit
        syscall

       .data
msg:   .asciiz    "The answer is:  "
```

Trace the code

mflo = ?
Mfhi = ?
$t3 = ?
$t4 = ?

Divide

```
The answer is:  3
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	3
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	24
$t1	9	8
$t2	10	0
$t3	11	3
$t4	12	0

Polynomial evaluation

Example-10

Using MIPS Assembly evaluate the polynomial: $2x^3 + 4$, for $x = 2$

with: `x = $t0`.

5 min

$2x^3 + 4$, for $x = 2$

```
        .text
        .globl main
main:
        li    $t0, 2
        mul   $t1, $t0, $t0        # $t0*$t0=2*2=4
        mul   $t1, $t1, $t0        # $t0*$t0*$t0=2*2*2=8
        mul   $t1, $t1, 2          # $t0*$t0*$t0*2=2*2*2*2=16
        addi  $t2, $t1, 4          # $t0*$t0*$t0*2+4=2*2*2*2+4=20

        li,   $v0, 10
        syscall
```

$t2 = ?

$$2*x^3 + 4 = 2*2^3 + 4 = 16 + 4 = 20$$

$2x^3 + 4$, for $x = 2$

```
        .text
        .globl main
main:
        li    $t0, 2
        mul   $t1, $t0, $t0
        mul   $t1, $t1, $t0
        mul   $t1, $t1, 2
        addi  $t2, $t1, 4

        li    $v0, 10
        syscall
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	2
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	2
$t1	9	16
$t2	10	20
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194336
hi		0
lo		16

$t2 = 20

$2 * x^3 + 4 = 2 * 2^3 + 4 = 16 + 4 = 20$

Another way

```
        .text
        .globl main
main:
        li   $t0, 2
        li   $t1, 4
        mul  $t2, $t0, $t0
        mul  $t3, $t2, $t0
        mul  $t4, $t3, $t0
        add  $t5, $t4, $t1

        li   $v0, 10
        syscall
```

Another way … input any number

```
        # Rich Barber (F'2018)
        .text
        .globl main
main:
        li      $v0, 5
        syscall
        move    $t0, $v0
        move    $t1, $t0
        mul     $t0, $t0, $t1
        mul     $t0, $t0, $t1
        mul     $t0, $t0, 2
        addi    $t0, $t0, 4
        move    $a0, $t0
        li      $v0, 1
        syscall
        li      $v0, 10
        syscall
```

Example-11

The implemented function (formula) is: ?

```
        .text
        .globl main
main:
        li    $t0,3
        li    $t1,2
        li    $t2,2
        mul   $t3,$t2,$t2
        mul   $t4,$t0,$t3
        mul   $t5,$t1,$t2

        add   $t6,$t4,$t5
        addi  $t7,$t6,1

        li    $v0, 10
        syscall
```

$t7 = ?

5 min

$3x^2 + 2x + 1$, for $x = 2$

```
        .text
        .globl main
main:
        li   $t0,3
        li   $t1,2
        li   $t2,2
        mul  $t3,$t2,$t2
        mul  $t4,$t0,$t3
        mul  $t5,$t1,$t2

        add  $t6,$t4,$t5
        addi $t7,$t6,1

        li   $v0, 10
        syscall
```

$$ax^2 + bx + c = 0$$

A General Quadratic Equation

$t7 = 17