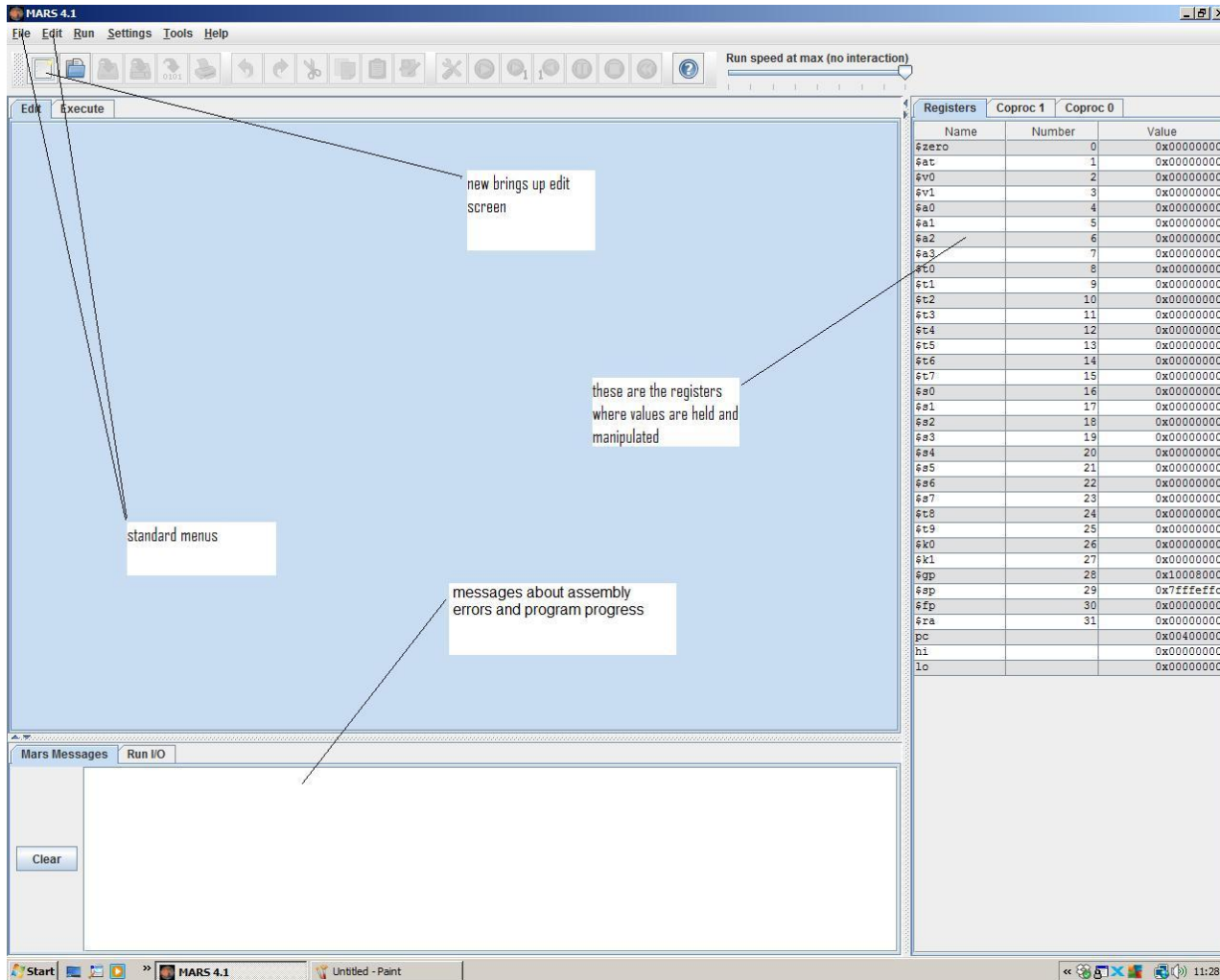


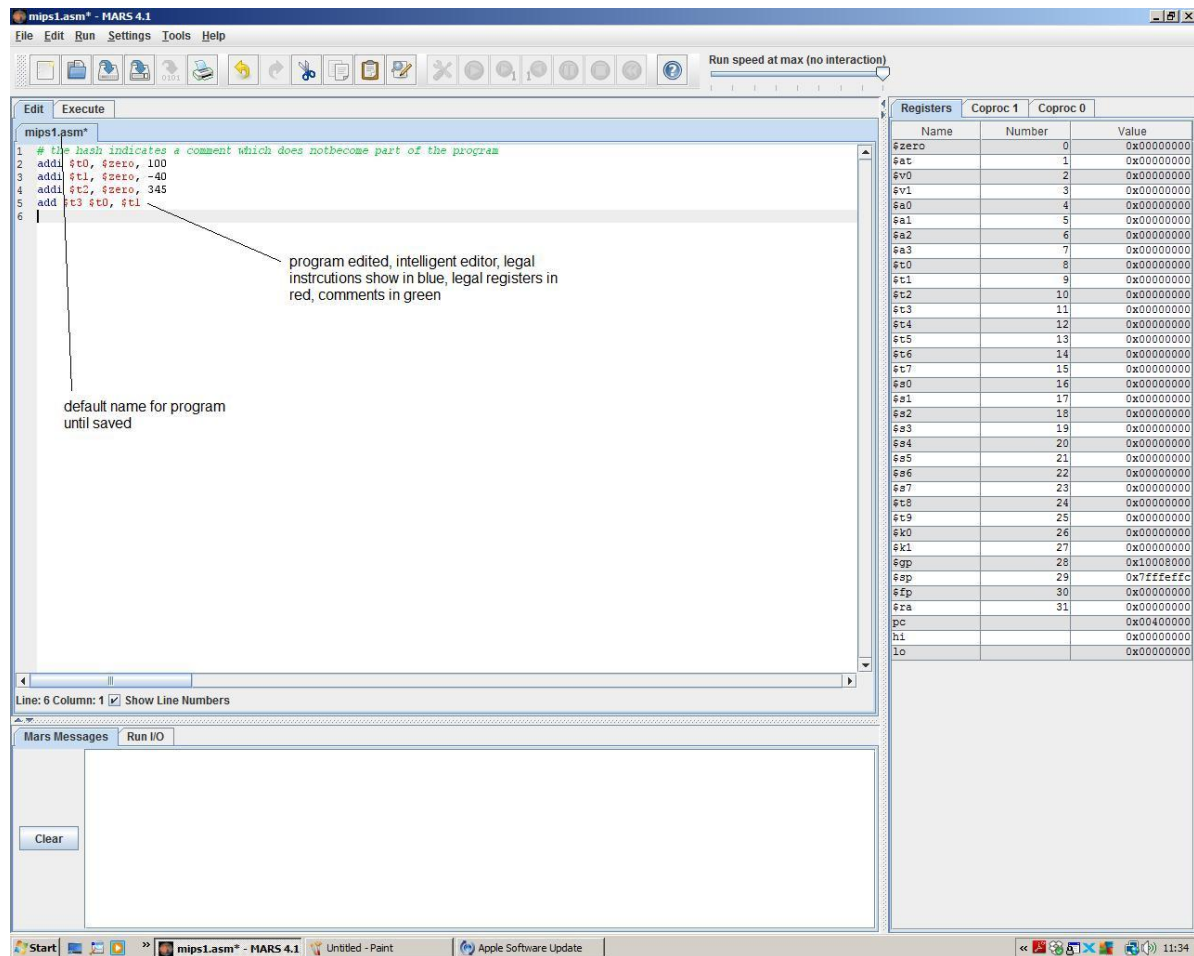
Using MARS

- Mars is a mips simulation
- A processor is defined by its register set and the instruction set which manipulates them
- Mars.jar is a java simulation of the Mips processor
- Download from moodle and double-click the file to run

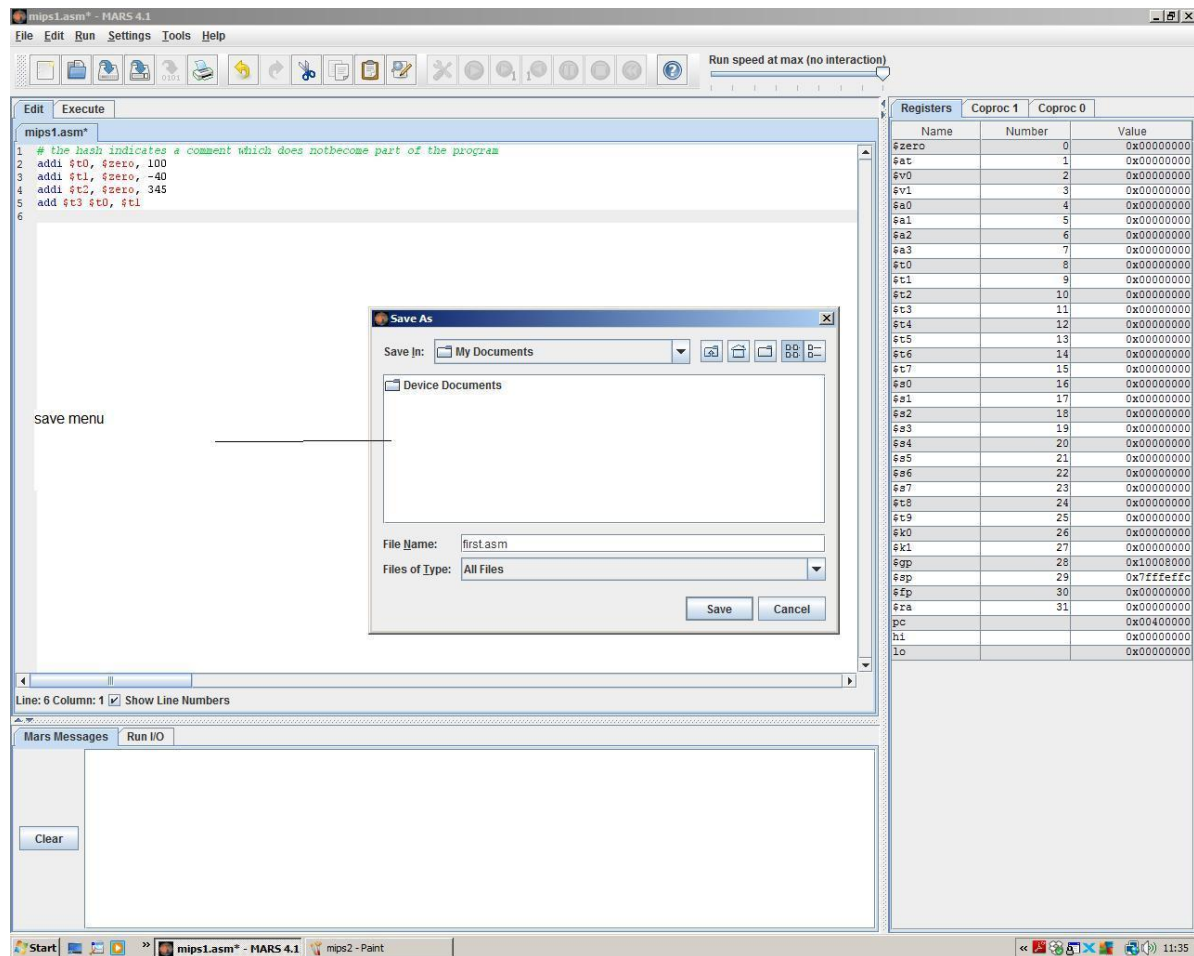
Opening screen



Edit screen



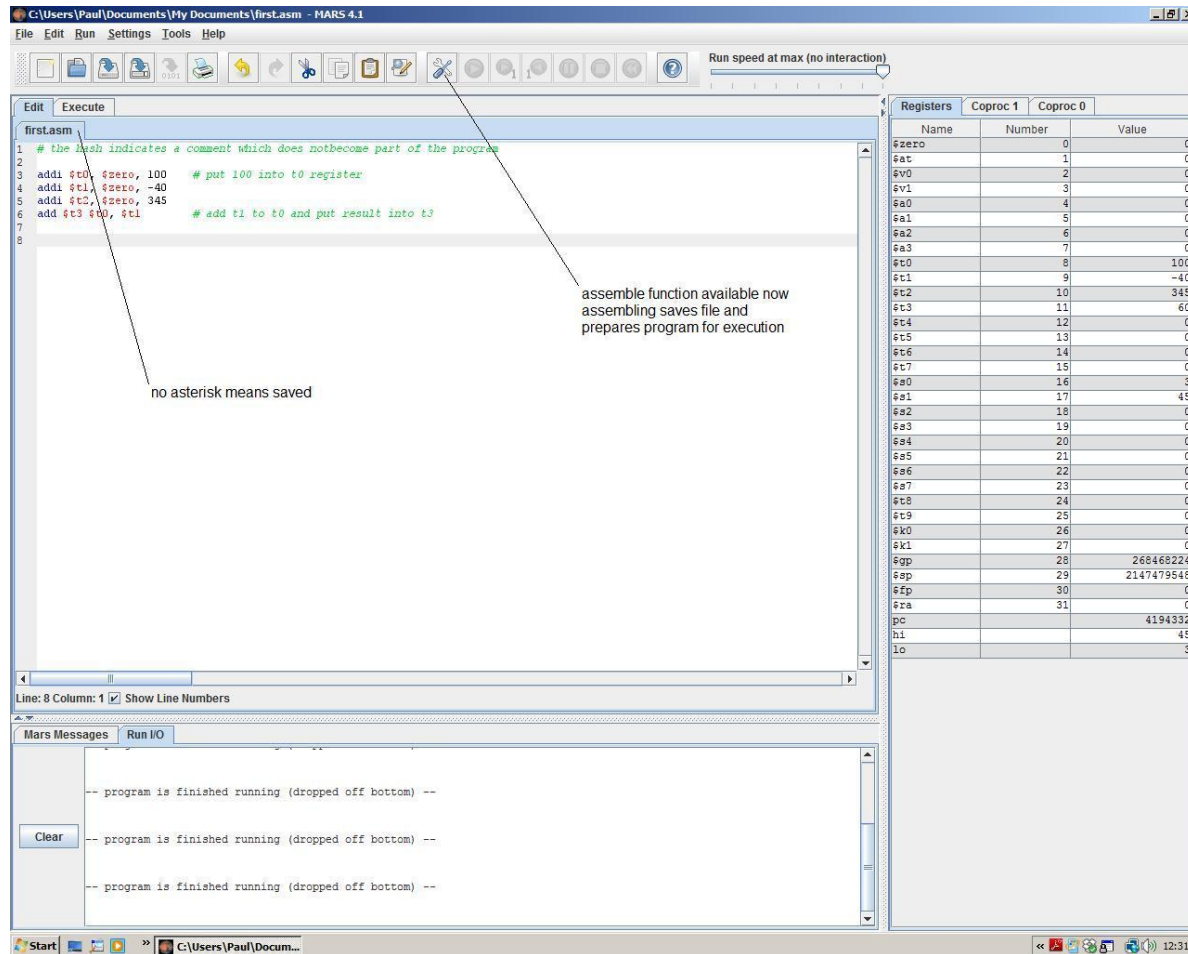
Save file



Saving

- Saving allows assembly and execution to take place
- Assembling – checks that program consists of legal operations
- Assembling – does not check that your program does anything sensible

Once saved



Execution screen

The screenshot displays the MARS 4.1 MIPS assembler simulator. The main window is titled "C:\Users\Paul\Documents\My Documents\first.asm - MARS 4.1". The menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains icons for file operations, editing, and execution. The "Run" button is highlighted with a green circle, and a tooltip indicates "Run speed at max (no interaction)".

The "Text Segment" panel shows the following code:

Bkpt	Address	Code	Basic	Source
0x00400000	0x20080064	addi \$t0,\$zero,100	2: addi \$t0, \$zero, 100	
0x00400004	0x2009ffdc	addi \$t1,\$zero,-40	3: addi \$t1, \$zero, -40	
0x00400008	0x200a0159	addi \$t2,\$zero,345	4: addi \$t2, \$zero, 345	
0x0040000c	0x01095520	add \$t3,\$t0,\$t1	5: add \$t3 \$t0, \$t1	

The "Data Segment" panel shows a table of memory addresses and their values in various offsets.

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

The "Registers" panel shows the state of MIPS registers:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

The "Mars Messages" panel shows the following messages:

```
Assembler: assembling C:\Users\Paul\Documents\My Documents\first.asm
Assembler: operation completed successfully.
```

Annotations in the image point to the following elements:

- execution window
- highlighted line is next to be executed
- execution options: run, single step, rewind to start
- register and data in hexadecimal

execution

- The execution screen allows for different modes
 - Run through – speed can be altered
 - Step through – line by line
 - Rewind to start
- Once executing other modes appear
 - Pause
 - Step back one
 - stop

Executing program

stepping through

option to step back now

first 3 lines have been executed and values placed in indicated registers

switch to ascii values

Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x20080064	addi \$t0, \$zero, 100	2: addi \$t0, \$zero, 100
	4194308	0x2009ffde	addi \$t1, \$zero, -40	3: addi \$t1, \$zero, -40
	4194312	0x200a0159	addi \$t2, \$zero, 345	4: addi \$t2, \$zero, 345
	4194316	0x01095520	add \$t1, \$t0, \$t2	5: add \$t1, \$t0, \$t1

Registers

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	100
\$t1	9	40
\$t2	10	345
\$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
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194316
hi		0
lo		0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501024	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501056	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501088	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501152	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501184	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501216	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501248	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501280	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501312	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501344	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501376	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Mars Messages

Run I/O

-- program is finished running (dropped off bottom) --

Reset: reset completed.

Clear

stuff

- Values appear in registers
- add, sub, mul, div – basic arithmetic ops
- mul, div – makes use of hi and lo regs
- Many other operations - look at help menu

help

C:\Users\Paul\Documents\My Documents\first.asm - MARS 4.1

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$t0	4	0
\$t1	5	0
\$t2	6	0
\$t3	7	0
\$t4	8	100
\$t5	9	-40
\$t6	10	345
\$t7	11	0
\$t8	12	0
\$t9	13	0
\$s0	14	0
\$s1	15	0
\$s2	16	0
\$s3	17	0
\$s4	18	0
\$s5	19	0
\$s6	20	0
\$s7	21	0
\$s8	22	0
\$s9	23	0
\$s10	24	0
\$s11	25	0
\$s12	26	0
\$s13	27	0
\$s14	28	268468224
\$s15	29	2147479548
\$s16	30	0
\$s17	31	0
\$s18		4194316
\$s19		0

Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x20080064	addi \$t0,\$zero,100	2: addi \$t0, \$zero, 100
	4194308	0x2009ffde	addi \$t1,\$t0,-40	3: addi \$t1, \$zero, -40
	4194312	0x200a0159	addi \$t2,\$t1,100	
	4194316	0x01095520	addi \$t3,\$t2,100	

MARS 4.1 Help

Operand Key for Example Instructions

- label, target any textual label
- \$t1, \$t2, \$t3 any integer register
- \$f2, \$f4, \$f6 even-numbered floating point register
- \$f0, \$f1, \$f3 any floating point register

Basic Instructions Extended (pseudo) Instructions Directives Syscalls Exceptions

abs.d \$f2,\$f4 Floating point absolute value double precision : Set \$f2 to absolute value of \$f4, double precision

abs.s \$f0,\$f1 Floating point absolute value single precision : Set \$f0 to absolute value of \$f1, single precision

add \$t1,\$t2,\$t3 Addition with overflow : set \$t1 to (\$t2 plus \$t3)

add.d \$f2,\$f4,\$f6 Floating point addition double precision : Set \$f2 to double-precision floating point sum of \$f4 and \$f6

add.s \$f0,\$f1,\$f3 Floating point addition single precision : Set \$f0 to single-precision floating point sum of \$f1 and \$f3

addui \$t1,\$t2,-100 Addition immediate unsigned without overflow : set \$t1 to (\$t2 plus signed 16-bit immediate)

addu \$t1,\$t2,\$t3 Addition unsigned without overflow : set \$t1 to (\$t2 plus \$t3), no overflow

and \$t1,\$t2,\$t3 Bitwise AND : Set \$t1 to bitwise AND of \$t2 and \$t3

andi \$t1,\$t2,100 Bitwise AND immediate : Set \$t1 to bitwise AND of \$t2 and zero-extended 16-bit immediate

bcif \$t1,label Branch if specified FP condition flag false (BCIF, not BCLF) : If Coprocessor 1 condition flag is false, branch to label

bcif \$t1,label Branch if specified FP condition flag true (BCIF, not BCLF) : If Coprocessor 1 condition flag is true, branch to label

bcif \$t1,label Branch if specified FP condition flag true (BCIF, not BCLF) : If Coprocessor 1 condition flag is true, branch to label

bcif \$t1,label Branch if specified FP condition flag true (BCIF, not BCLF) : If Coprocessor 1 condition flag is true, branch to label

beq \$t1,\$t2,label Branch if equal : Branch to statement at label's address if \$t1 and \$t2 are equal

bgez \$t1,label Branch if greater than or equal to zero : Branch to statement at label's address if \$t1 is greater than or equal to zero

bgez \$t1,label Branch if greater than or equal to zero and link : If \$t1 is greater than or equal to zero, branch to statement at label's address if \$t1 is greater than or equal to zero and link

bgez \$t1,label Branch if greater than or equal to zero and link : If \$t1 is greater than or equal to zero, branch to statement at label's address if \$t1 is greater than or equal to zero and link

blez \$t1,label Branch if less than or equal to zero : Branch to statement at label's address if \$t1 is less than or equal to zero

blez \$t1,label Branch if less than or equal to zero : Branch to statement at label's address if \$t1 is less than or equal to zero

Mars Messages Run I/O

-- program is finished running (dropped off bottom) --

Reset: reset completed.

Close

Start C:\Users\Paul\Docum... mips5 - Paint 11:40

div

The screenshot shows the MARS 4.1 MIPS assembler simulator. The main window displays assembly code for a file named `first.asm`. The code includes comments explaining each instruction. A text annotation with arrows points to the `mfhi $s0` and `mflo $s1` instructions, stating: "certain instructions use special registers hi and lo".

```
1 # the hash indicates a comment which does not become part of the program
2
3 addi $t0, $zero, 100 # put 100 into t0 register
4 addi $t1, $zero, -40
5 addi $t2, $zero, 345
6 add $t3, $t0, $t1 # add t1 to t0 and put result into t3
7 div $t2, $t0 # now look at hi and lo
8 mflo $s0 # access the quotient from the division
9 mfhi $s1 # access the remainder from the division
10
```

Below the code editor, the "Mars Messages" window shows the execution log:

```
Go: running first.asm
Go: execution terminated by null instruction.
Assemble: assembling C:\Users\Paul\Documents\My Documents\first.asm
Assemble: operation completed successfully.
```

On the right side, the "Registers" window displays the state of the MIPS registers. The table below shows the values for the registers.

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

hi and lo

C:\Users\Paul\Documents\My Documents\first.asm - MARS 4.1

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

```
1 # the hash indicates a comment which does not become part of the program
2
3 addi $t0, $zero, 100 # put 100 into t0 register
4 addi $t1, $zero, -40
5 addi $t2, $zero, 345
6 add $t3, $t0, $t1 # add t1 to t0 and put result into t3
7 div $t2, $t0 # now look at hi and lo
8 mflo $s0 # access the quotient from the division
9 mghi $s1 # access the remainder from the division
10
```

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	100
\$t1	9	-40
\$t2	10	345
\$t3	11	60
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	3
\$s1	17	45
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194332
hi		45
lo		3

Line: 6 Column: 55 ☒ Show Line Numbers

Mars Messages Run I/O

-- program is finished running (dropped off bottom) --

Clear

-- program is finished running (dropped off bottom) --

-- program is finished running (dropped off bottom) --

Start C:\Users\Paul\Docum... mps7 - Paint 11:56

Diagram illustrating the relationship between the assembly code and the register values:

- Line 6: `div $t2, $t0` (now look at hi and lo) points to the `hi` register value (45).
- Line 8: `mflo $s0` (access the quotient from the division) points to the `lo` register value (3).
- Line 9: `mghi $s1` (access the remainder from the division) points to the `hi` register value (45).

Some info about registers

The screenshot displays the MARS 4.1 MIPS assembler simulator interface. The main window is divided into several sections:

- Text Segment:** A table showing assembly instructions with columns for Bkpt, Address, Code, Basic, and Source. The instructions are:
 - 4194304: `addi $t0, $zero, 100` (Basic: `3: addi $t0, $zero, 100`; Source: `# put 100 into t0 register`)
 - 4194308: `addi $t1, $zero, -40` (Basic: `4: addi $t1, $zero, -40`; Source: `# add t1 to t0 and put result into t3`)
 - 4194312: `addi $t2, $zero, 345` (Basic: `5: addi $t2, $zero, 345`; Source: `# now look at hi and lo`)
 - 4194316: `div $t0, $t1` (Basic: `6: div $t0, $t1`; Source: `# access the quotient from the division`)
 - 4194320: `mflo $f0` (Basic: `7: mflo $f0`; Source: `# access the remainder from the division`)
 - 4194324: `mfihi $s1` (Basic: `8: mfihi $s1`; Source: `# access the remainder from the division`)
 - 4194328: `mfihi $s1` (Basic: `9: mfihi $s1`; Source: `# access the remainder from the division`)
- Registers:** A table on the right showing the state of registers. The registers are divided into Coproc 1 and Coproc 0. The registers shown are:
 - `$zero`: 0
 - `$at`: 1
 - `$v0`: 2
 - `$v1`: 3
 - `$a0`: 4
 - `$a1`: 5
 - `$a2`: 6
 - `$a3`: 7
 - `$t0`: 8
 - `$t1`: 9
 - `$t2`: 10
 - `$t3`: 11
 - `$t4`: 12
 - `$t5`: 13
 - `$t6`: 14
 - `$t7`: 15
 - `$s0`: 16
 - `$s1`: 17
 - `$s2`: 18
 - `$s3`: 19
 - `$s4`: 20
 - `$s5`: 21
 - `$s6`: 22
 - `$s7`: 23
 - `$t8`: 24
 - `$t9`: 25
 - `$k0`: 26
 - `$k1`: 27
 - `$gp`: 28 (Value: 268468224)
 - `$sp`: 29 (Value: 2147479548)
 - `$fp`: 30
 - `$ra`: 31
 - `pc`: 4194304
 - `hi`: 0
 - `lo`: 0
- Data Segment:** A table showing memory addresses and their values. The values are represented as hexadecimal digits (0-9, A-F).
- Mars Messages:** A log window at the bottom showing the execution process:
 - Go: running first.asm
 - Go: execution terminated by null instruction.
 - Clear
 - Assemble: assembling C:\Users\Paul\Documents\My Documents\first.asm
 - Assemble: operation completed successfully.