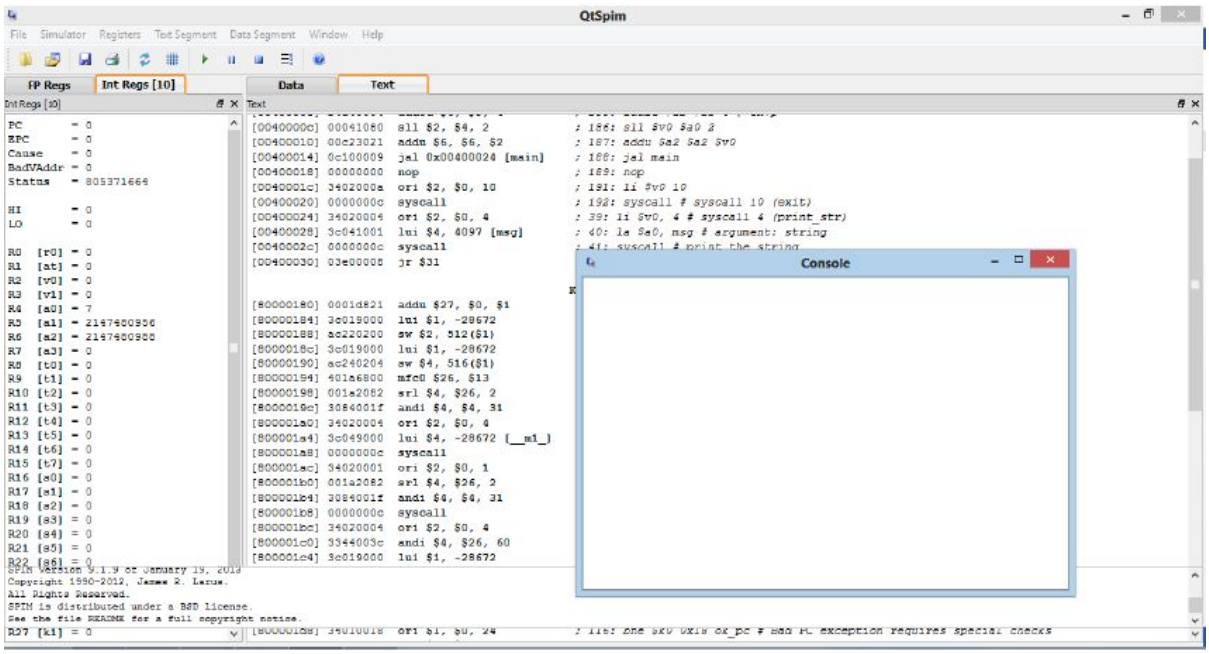
Qtspim Introduction

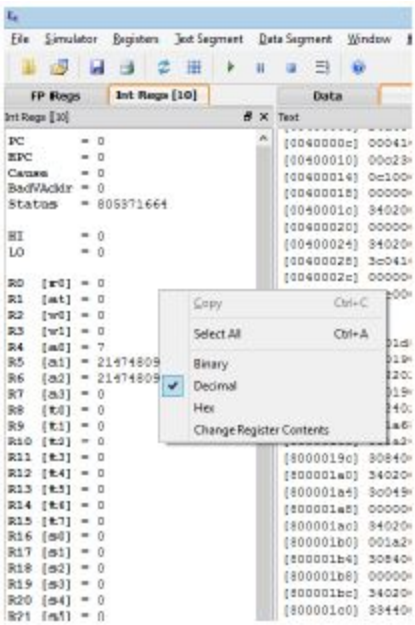
written by: Ferdinand - Computer Science 2014

1. Interface



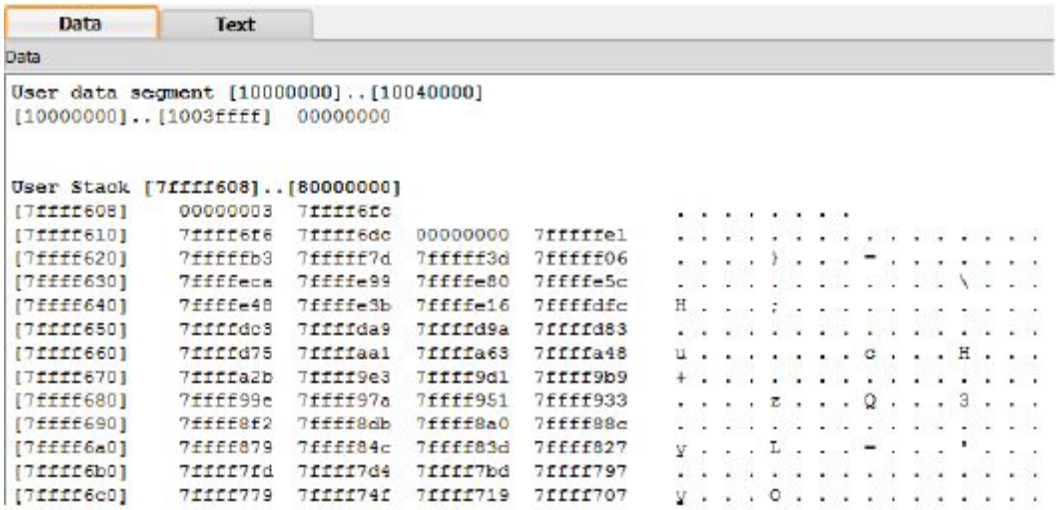
Picture 1: Qtspim Display[6]

QtSpim has two views. QtSpim first view displays all information related line of instructions, registers and data. The second display is in the form of a console that displays the output of the program.



The left side is a display that provides information register. It displays a list of registers and values contained in the registers. MIPS has a register that stores integer values and floating point values. The value in the register can be displayed in binary format, decimal, and hexadecimal (Right-click).

Picture 2: Left Side of Qtspim[6]



Picture 3: Right Side of Qtspim[6]

The right side of the display has two tabs: Data and Text. Text displays a line of instructions that have been loaded previously in QtSpim. Data shows the address of the memory used to store data and their values.

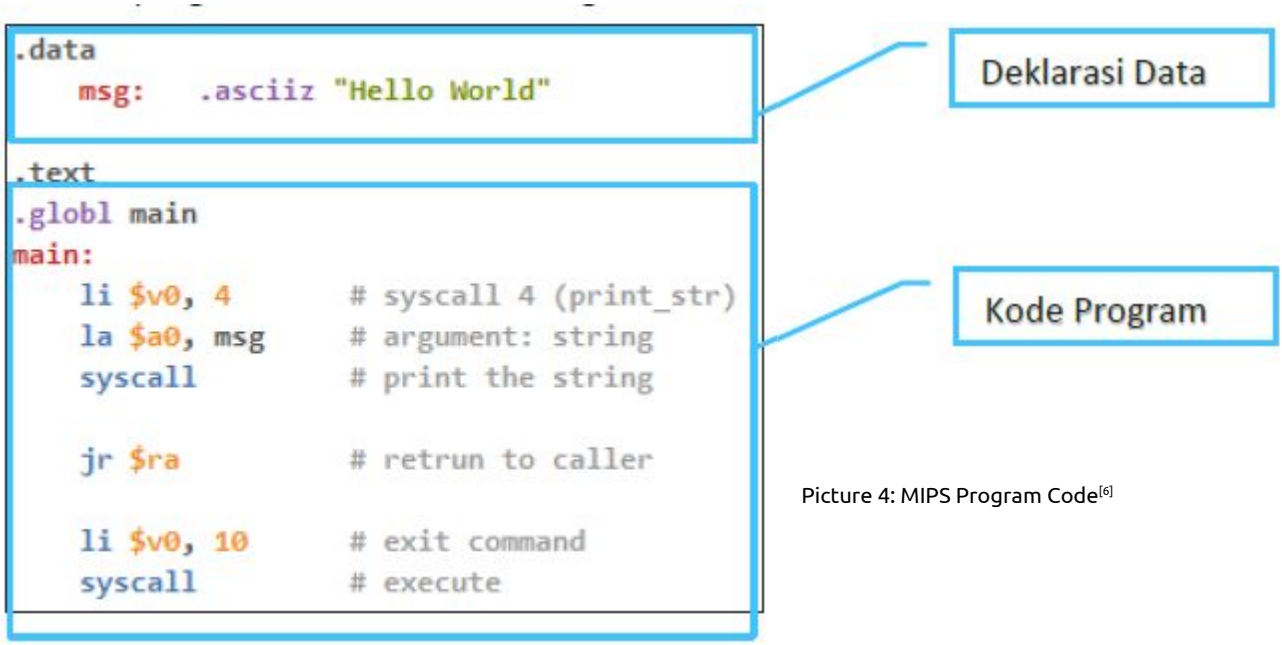
2. Register

Register Number	Conventional Name	Usage
$0	$zero	Hard-wired to 0
$1	$at	Reserved for pseudo-instructions
$2 - $3	$v0, $v1	Return values from functions
$4 - $7	$a0 - $a3	Arguments to functions - not preserved by subprograms
$8 - $15	$t0 - $t7	Temporary data, not preserved by subprograms
$16 - $23	$s0 - $s7	Saved registers, preserved by subprograms
$24 - $25	$t8 - $t9	More temporary registers, not preserved by subprograms
$26 - $27	$k0 - $k1	Reserved for kernel. Do not use.
$28	$gp	Global Area Pointer (base of global data segment)
$29	$sp	Stack Pointer
$30	$fp	Frame Pointer
$31	$ra	Return Address
$f0 - $f3	-	Floating point return values
$f4 - $f10	-	Temporary registers, not preserved by subprograms
$f12 - $f14	-	First two arguments to subprograms, not preserved by subprograms
$f16 - $f18	-	More temporary registers, not preserved by subprograms
$f20 - $f31	-	Saved registers, preserved by subprograms

Table 1: Register in MIPS[7]

There are also special registers, Lo and Hi is used to store the value of multiplication and division.

3. Program Structure



Picture 4: MIPS Program Code[6]

.data is a mark for declaration of data section. In this section, the name of the variables used in the program was written and will be allocated in the memory. Variables are declared in this section may consist of one or more variables.

.text is a mark for program code. Rows of instruction to be executed is written bellow this section.

Syntax **main**: is called label. Label is a name for a place to branch to.

IMPORTANT!

Always end your program with:

```
li $v0, 10 # exit command
syscall # execute
```

4. Data Declaration

How to declare a data:

```
[varName]: [.type] [value]
```

Example:

```
msg: .asciiz "Hello World"
```

4.1. Data Type

Data Type	Value	Explanation
.ascii	str	Store string in memory without null terminator (\ n)
.asciiz	str	Store string in memory with null terminator (\ n)
.byte	b1, …, bn	Store n bytes in memory. Can be written in base 10, or hex. Each value is separated by a comma (,)
.halfword	h1, …, hn	Store n halfword in memory. Can be written in base 10, or hex. Each value is separated by a comma (,)
.word	w1, …, wn	Store n word in memory. Can be written in base 10, or hex. Each value is separated by a comma (,)
.space	numBytes	Reserves numBytes of space in memory.

Table 2: Data Type in MIPS

5. Instruction

Arithmetic and Logical Instruction

Instruction	Syntax	Penjelasan
add	$d, $s, $t	$d = $s + $t
addi	$d, $s, i	$d = $s + $i
and	$d, $s, $t	$d = $s AND $t
andi	$d, $s, i	$d = $s AND i
div	$s, $t	$s / $t, $LO = quotient, $HI = remainder
mult	$s, $t	$LO = $s * $t
or	$d, $s, $t	$d = $s OR $t

ori	$d, $s, i	$d = $s OR i
sll	$d, $t, a	$d = $t is left shifted a times
srl	$d, $t, a	$d = $t is right shifted a times
sub	$d, $s, $t	$d = $s - $t

Branch Instruction

Instruction	Syntax	Penjelasan
beq	$s, $t, label	Go to label if $s = $t
bgtz	$s, label	Go to label if $s > 0
blez	$s, label	Go to label if $s <= 0
bne	$s, $t, label	Go to label if $s != $t

Jump Instruction

Instruction	Syntax	Penjelasan
j	label	Jump to label
jal	label	Jump to label, $31 = return address
jr	$ra	Jump to address that $ra stored

Load Instruction

Instruction	Syntax	Penjelasan
lb	$t, i ($s)	$t = MEM[$s + i]. A byte is loaded into a register from the specified address.
lw	$t, i ($s)	$t = MEM[$s + i]. A word is loaded into a register from the specified address.

Store Instruction

Instruction	Syntax	Penjelasan
sb	$t, i ($s)	MEM[$s + offset] = (0xff & $t). The least significant byte of $t is stored at the specified address.
sw	$t, i ($s)	MEM[$s + i] = $t. The contents of $t is stored at the specified address

Data Movement Instruction

Instruction	Syntax	Operation
`mfhi`	`$d`	`$d = $HI`
`mflo`	`$d`	`$d = $LO`

More details can be found at:

- http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html
- http://alumni.cs.ucr.edu/~vladimir/cs161/mips.html

6. Syscall (Input/Output)

For input and output, Qtspim uses ***syscall***. Syscall read the code at the register $v0. Each code has a different service.

Service	Kode	Argumen	Hasil
`print integer`	1	$a0 = value	(none)
`print float`	2	$f12 = float value	(none)
`print double`	3	$f12 = double value	(none)
`print string`	4	$a0 = address of string	(none)
`read integer`	5	(none)	Disimpan di $v0
`read float`	6	(none)	Disimpan di $f0
`read double`	7	(none)	Disimpan di $f0
`read string`	8	$a0 = address string mau disimpan $a1 = jumlah karakter yang mau dibaca + 1	(none)
`memory allocation`	9	$a0 = banyaknya storage dalam byte yang diinginkan	$v0 = alamat block
`exit (end of program)`	10	(none)	(none)
`print character`	11	$a0 = integer	(none)
`read character`	12	(none)	Char disimpan di $v0

Tabel 3: Syscall [2]

Reference:

[1]http://courses.cs.washington.edu/courses/cse410/99au/lectures/Lecture-09-29/sld022.htm
[2] http://students.cs.tamu.edu/tanzir/csce350/reference/syscalls.html
[3] https://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Mips/dataseg.html
[4] http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html
[5] http://alumni.cs.ucr.edu/~vladimir/cs161/mips.html
[6] Priyandhana A. (2014). Tutorial I MIPS: Pengenalan QtSpim, register, struktur program, deklarasi data, instruksi dasar, I/O, penulisan dokumentasi, latihan. *Dasar-Dasar Arsitektur Komputer, Fakultas Ilmu Komputer Universitas Indonesia.*
[7] http://www.cs.uwm.edu/classes/cs315/Bacon/Lecture/HTML/ch05s03.html

Exercise

Yolo is a ***single*** man. He don't want to celebrate valentine alone. There are three girl that Yolo wants to spend valentine with: *Silvi*, *Veronica*, and *Farhati*. He wants to celebrate Valentine with a woman that suits him. After consulting with the experts of love, Mr. V, Yolo got a formula to measure the suitability based on date of birth. The formula is:

```
((Male Date of Birth + Female Date of Birth)*3) - ((Female Date of Birth + Male Date of
                                        Birth)/4)
```

If the result of the calculation is an **odd** number, it means that they **fit** each other.

Make the assembly program to help Yolo met her match.

Specification:

- Input:
  - Male Date of Birth
  - Female Date of Birth
- Output:
  - Calculation result
  - String to show if they match or not. (exp. `MATCH`)
- Save the file with format: `[Nama]_[NPM]_[Kelas].s`