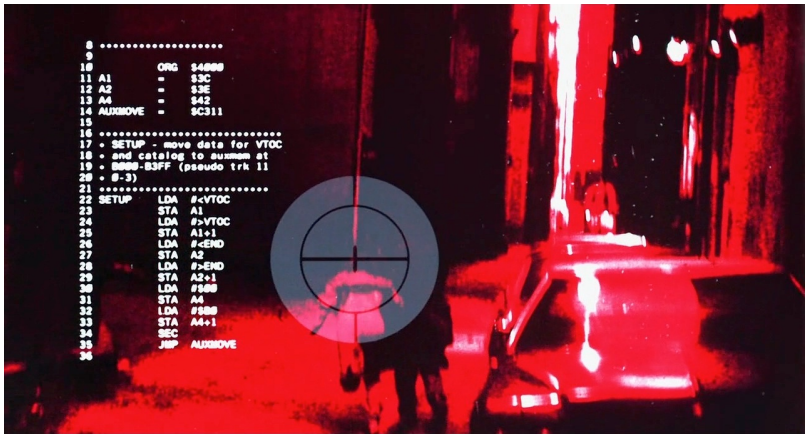
MIPS and SPIM tutorial



(Did you know that the T-800 Terminator runs on 6502 Assembly? You too can learn to program in assembly!)

To be able to understand how to program the MIPS processor using the MIPS assembly language, there is no better way than start coding yourself. A good way to learn is learning from examples.

Part Zero

Assembly programming is closely related to the hardware. Details of the hardware, such as registers, are not visible to the programmer when using other more high level programing languages.

In this part of the tutorial you will learn about the ALU, the registers and the memory. To better understand the MIPS assembly language (design) this knowledge is needed.

Part One

To execute your MIPS programs you will use the MIPS simulator SPIM. In the first part of the tutorial you will learn how to use SPIM and get started with the MIPS assembly language. The instructions and directives introduced in this part of the tutorial are:

Instruction	Meaning
add	Addition
addi	Add Immediate
seq	Set Equal
jal	Jump and Link
jr	Jump Register
Directive	Meaning
.text	Treat the following code as instructions and place the result in the text segment

Download the following MIPS assembly program: first_try.s

Open up *Part One: Your first assembly program* (pdf) and follow the instructions.

An alternative to SPIM is MARS. In look and feel MARS is probably a little more "modern". Before you start using MARS you should know about a few differences between SPIM and MARS

Part Two

In this part you will learn how to:

- Store data from a register to a location (address) in memory.
- Load data from a memory location to a register
- Translate a label into an address.

The instructions and directives introduced in this part of the tutorial are:

Instruction	Meaning
lw	Load Word
sw	Store Word
la	Load Address
Directive	**Meaning**
.data	Treat the following code as data and place the result in the data segment
.word	Store a word (four bytes)
.space	Reserve space in the data segment

Download the following MIPS assembly program: load_and_store.s

Open up *Part Two: Loads and stores* (pdf) and follow the instructions.

Part Three

In this part you will learn:

- To structure your programs using **subroutines**
- About the MIPS calling convention, a.k.a, t**he register convention**
- More about **Jump and Link**
- About the stack (**push & pop**)

The instructions and directives introduced in this part of the tutorial are:

Instruction	Meaning
li	Load Immediate
nop	No Operation, an instruction that does nothing.

Download the following MIPS assembly program: subroutines.s

Open up *Part Three: Subroutines* (pdf) and follow the instructions.

Part Four

In this part you will learn:

- How to represent and store **strings** and **characters** in the data segment.
- About **the ASCII system** used to encode characters as 8 bit numbers.location to a register
- About **Little Endian** and **Big Endian** (byte order)
- How to use **syscalls** in SPIM to print integers and strings
- To how to use branch instructions to construct **loops**
- More about how to write **good comments** to document your code
- How to construct **If-Then-Else** expressions
- About **arrays**

The instructions and directives introduced in this part of the tutorial are:

Instruction	Meaning
lb	Load Byte
beq	Branch Equal
j	Unconditional Jump
bne	Branch Not Equal
sll	Shift Left Logical
Directive	**Meaning**
.asciiz	Stores a NUL terminated string in the data segment
.word	Store a word (four bytes)
.space	Reserve space in the data segment

Download the following MIPS assembly programs:

- string.s
- print_string_and_integer.s
- loop.s
- if_then_else.s
- array.s
- array_of_strings.s

Open up *Part Four: Strings, Loops, Ifs, Arrays* (pdf) and follow the instructions.

Part Five

In this part you will learn about:

- **Exceptions -** internal and synchrounus errors in a program.
- **Interrupts-** external and asynchrounus events.
- **Memory mappad I/O**
- **Polled I/O, Interrupt driven I/O and DMA.**

The instructions and directives introduced in this part of the tutorial are:

Instruction	Meaning
lui	Load Upper Immediate

ori	Or Immediatel
mfc0	Move From Coprocessor 0
srl	Shift Right Logical
andi	And Immediate

Directive	**Meaning**
.ktext	The assembler will put the resulting instructions in the kernel text segment.
.kdata	The assembler will put the resulting data in the kernel data segment.

Open up *Part Five: Exceptions and Interrupts* (pdf) and follow the instructions.

Revisions

20080901 Karl Marklund
20140901 Germán Ceballos