

NOTES :

Before using the program please test each instruction beforehand so you d be sure every thing works as you intend it to be

User interface :

Mips Compiler

REG0 trash

REG1 trash

REG2 trash

REG3 trash

REG4 trash

REG5 trash

REG6 trash

REG7 trash

inputReg trash mem

outputReg trash mem

program counter 0

Run

LineByLine

print mem binary

MEM00 trash mem

MEM01 trash mem

MEM02 trash mem

MEM03 trash mem

MEM04 trash mem

MEM05 trash mem

MEM06 trash mem

MEM07 trash mem

MEM08 trash mem

MEM09 trash mem

MEM10 trash mem

MEM11 trash mem

MEM12 trash mem

MEM13 trash mem

MEM14 trash mem

MEM15 trash mem

pause at

Continue

Restart

- app.txt
- bin.txt
- hex.txt
- input.txt
- INSTRUCTION SET.png
- mem.txt
- msvcp140.dll
- notes.docx
- Output.txt
- Project1.exe
- vcruntime140.dll

The program reads your application from app.txt

Press restart then run to run the entire program or line by line to ....

Binary prints your code in binary and hex , print memory prints your mem in hex values inside the file mem.txt

Pause at can pause the program when it reaches a certain line

(pause takes more time in execution so be patient )

A screenshot of a user interface element. It consists of a light gray rectangular container. Inside the container, on the left, is a button with the text "pause at" in a dark font. To the right of the button is a text input field with a blue border, containing the number "5".

Continue continues the program from where u paused

All user interface values are hexadecimal

---

## E) Instruction Description

Assembly Format	Description
ADD $R_d, R_s, R_t$	$R_d \leftarrow [R_s] + [R_t]; PC \leftarrow [PC] + 2$
SUB $R_d, R_s, R_t$	$R_d \leftarrow [R_s] - [R_t]; PC \leftarrow [PC] + 2$
AND $R_d, R_s, R_t$	$R_d \leftarrow [R_s] \text{ AND } [R_t]; PC \leftarrow [PC] + 2$
OR $R_d, R_s, R_t$	$R_d \leftarrow [R_s] \text{ OR } [R_t]; PC \leftarrow [PC] + 2$
NOR $R_d, R_s, R_t$	$R_d \leftarrow [R_s] \text{ NOR } [R_t]; PC \leftarrow [PC] + 2$
XOR $R_d, R_s, R_t$	$R_d \leftarrow [R_s] \text{ XOR } [R_t]; PC \leftarrow [PC] + 2$
SLL $R_d, R_s$	$R_d \leftarrow [R_s] \ll 1; PC \leftarrow [PC] + 2$
SRL $R_d, R_s$	$R_d \leftarrow [R_s] \gg 1; PC \leftarrow [PC] + 2$
ADDI $R_t, R_s, \text{immediate}$	$R_t \leftarrow [R_s] + ([I_5]^{10} \parallel [I_{5..0}]); PC \leftarrow [PC] + 2$
ANDI $R_t, R_s, \text{immediate}$	$R_t \leftarrow [R_s] \text{ AND } (0^{10} \parallel [I_{5..0}]); PC \leftarrow [PC] + 2$
ORI $R_t, R_s, \text{immediate}$	$R_t \leftarrow [R_s] \text{ OR } (0^{10} \parallel [I_{5..0}]); PC \leftarrow [PC] + 2$
LW $R_t, R_s$	$R_t \leftarrow M([R_s] + [I_5]^{10} \parallel [I_{5..0}]); PC \leftarrow [PC] + 2$
SW $R_t, R_s$	$M([R_s] + [I_5]^{10} \parallel [I_{5..0}]) \leftarrow [R_t]; PC \leftarrow [PC] + 2$
J jump_target	$PC \leftarrow [PC_{15..13}] \parallel [I_{11..0}] \parallel 0$
JAL jump_target	$R_7 \leftarrow [PC] + 2; PC \leftarrow [PC_{15..13}] \parallel [I_{11..0}] \parallel 0$
JR $R_s$	$PC \leftarrow [R_s]$
BEQ $R_s, R_t, \text{offset}$	if $[R_s] = [R_t]$ then $PC \leftarrow [PC] + 2 + ([I_5]^9 \parallel [I_{5..0}] \parallel 0)$ else $PC \leftarrow [PC] + 2$
BNE $R_s, R_t, \text{offset}$	if $[R_s] \neq [R_t]$ then $PC \leftarrow [PC] + 2 + ([I_5]^9 \parallel [I_{5..0}] \parallel 0)$ else $PC \leftarrow [PC] + 2$
IN $R_d$	$R_d \leftarrow \text{IN\_PORT}; PC \leftarrow [PC] + 2$
OUT $R_s$	$\text{OUT\_PORT} \leftarrow [R_s]; PC \leftarrow [PC] + 2$
NOP	$PC \leftarrow [PC] + 2$
HLT	-

Please note we erased the offset in LW and SW

## how to write instructions

\*All lines have to be capital and all numbers have to be decimal

Example: ORI R6, R6, 15

\*Maximum value for immediate numbers inside the instruction is 31 ; its supposed to be 63 but it caused some problems so test it yourself before going over 31

\*immediate value for 2 is 02 not 2 and zero is 00 and so on , and ofcourse immediate value is decimal

\*DO NOT LEAVE UNNECESSARY SPACES preferably no spaces at all

Jump :

All labels has to start with a small x not X and end with ":"

Example :

J xnameOfYourLabel < no ":" when using the label

.  
.   
.   
.

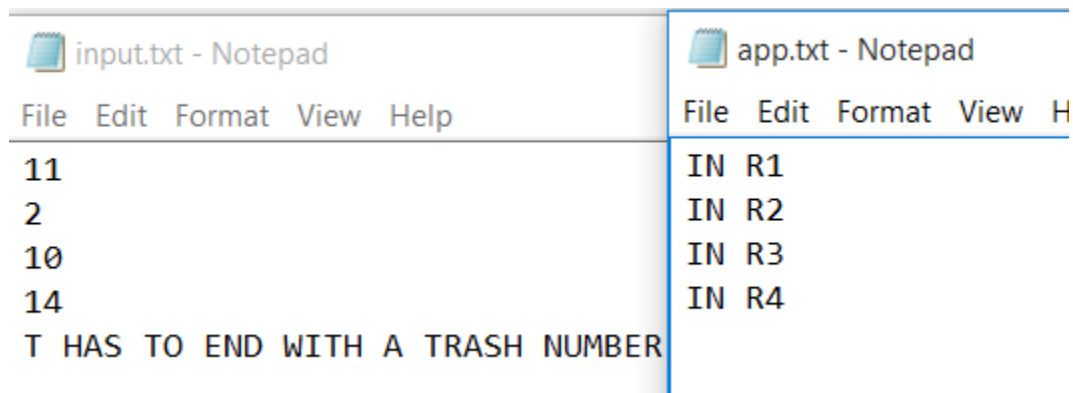
xnameOfYourLabel: ORI R6, R6, 15 → do not leave the label without instruction

OUT :

Puts a certain register on the output pins so we saved every out inside the out.txt and all the numbers are in hexa

IN :

Takes input from the input pins so you will have to specify the value each time you need it inside the input.txt file



ALL Input are decimal values

\*Memory:

As for memory file it contains only the memories that has a non zero value

Mem has  $2^{16}$  adress that is xFFFF as specified in the instruction sheet

If the program crashes that means you made an infinite loop