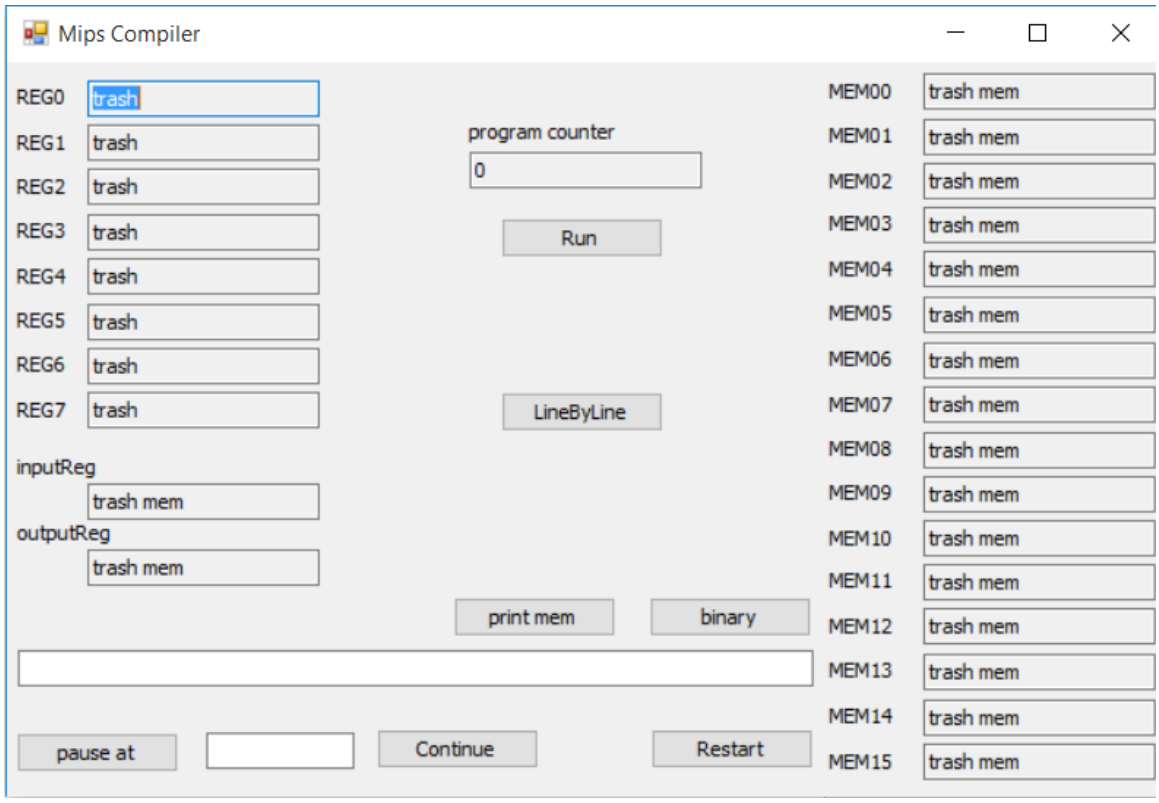
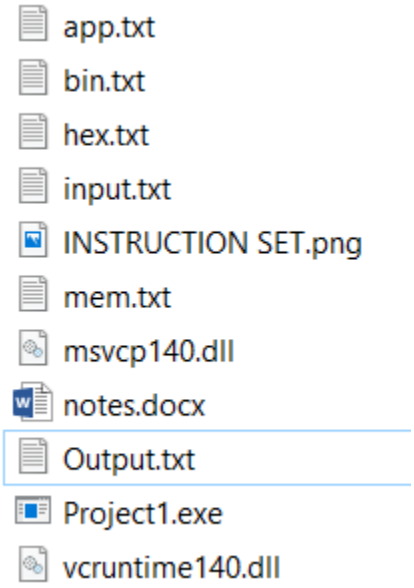
NOTES :

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 :





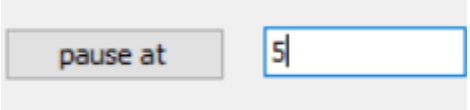
The program reads your application from app.txt

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 paitent)



Continue continues the program from where u paused

All user interface values are hexdecimal

E) Instruction Description

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

Please note we eraesed the offset in LW and SW

how to write instructions

*All lines have to be capital and all numbers have to be decimel

Example: ORI R6, R6, 15

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

*imidiate value for 2 is 02 not 2 and zero is 00 and so on , and ofcorse imidiate calue is decimal

*DO NOT LEAVE UNNECASRY 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

.

.

.

.

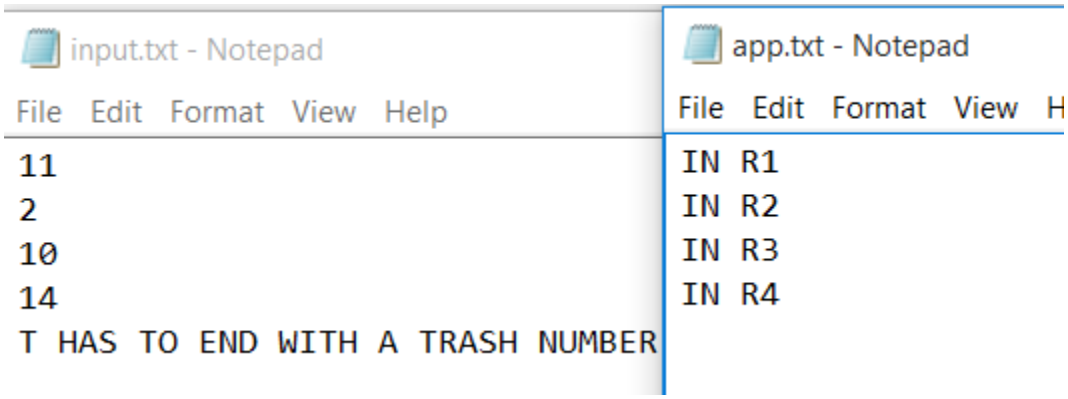
xnameOfYourLabel: ORI R6, R6, 15 → do not leave the label without instruction

OUT :

Puts a certain regester 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