**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**🙚🟃🙘**

**BÁO CÁO FINAL PROJECT**

**MÔN KIẾN TRÚC MÁY TÍNH**

**ĐỀ TÀI SỐ 1**

Giáo viên hướng dẫn : **ThS. Nguyễn Đức Tiến**

Sinh viên:

**Nguyễn Hữu Sơn – MSSV: 20111948**

**Lớp : 7C – K57 Việt Nhật**

*Hà Nội 5/2015*

Mục lục

\* \*

\*

1. **Đề bài**  Tr.4
2. **Phân tích cách làm** Tr.4
3. Thuật toán Tr.5
4. **Mã nguồn** Tr.7
5. **Kết quả chạy mô phỏng** Tr.15
6. Hiển thị kết quả
7. **Đề bài**

1. Sử dụng ngắt để thay đổi kịch bản led

Chương trình sau sẽ phát hiện trạng thái của các ngắt K1, K2 và 8 switch, từ đó hiển thị đèn led IO sáng

theo kịch bản như bên dưới: (Kí hiệu ∘: led tắt, O: led sáng)

- Giá trị khởi tạo của 8 led, có thể sáng hoặt tắt, theo cùng trạng thái/giá trị của 8 switch. Chẳng hạn, nếu

cả 8 switch đều bật thì cả 8 đèn cũng đều sáng.

- Sau đó cả 8 leds sẽ nhấp nháy theo tốc độ qui định bởi External Timer. Tức là External Timer sẽ sinh ra

ngắt, từ đó làm các led này đảo ngược giá trị của chúng, từ sáng  tắt, tắt  sáng. ∘ O → O ∘ → ∘ O

- Tuy nhiên, nếu ngắt K1 được bấm, thì tất cả 4 đèn led cao sáng, 4 led thấp tắt. O O O O ∘ ∘ ∘ ∘. Duy trì

các đèn ở trạng thái như vậy trong 3 chu kì của External Timer rồi lại nhấp nháy lại như ban đầu.

- Tuy nhiên, nếu ngắt K2 được bấm, thì chỉ 4 đèn led giữa sáng. ∘ ∘ O O O O ∘ ∘. Duy trì các đèn ở trạng

thái như vậy trong 3 chu kì của External Timer rồi lại nhấp nháy lại như ban đầu.

1. **Phân tích cách làm**
2. **Thuật toán**

Xuất hiện ngắt => chuyển đến chương trình xử lí ngắt.

Đọc trạng thái của thanh ghi trạng thái của tín hiệu ngắt.  
- K1 có thể là 0x20 hoặc 0x22 => andi với 0x20 nếu trả về 1 tức là ngắt do K1

- K2 có thể là 0x10 hoặc 0x11 => andi với 0x10 nếu trả về 1 tức là ngắt do K2

- Các trường hợp còn lại là K3

- Nếu là k1, k2 thì lặp như yêu cầu 3 lần, sử dụng biến đếm số lần, biến này không reset khi thoát khỏi hàm xử lí ngắt.

- Khi đủ 3 lần thì reset biến đếm, tín hiệu đèn về trước khi lặp 3 lần.

- Đọc tín hiệu của switch vào ghi giá trị cần thiết vào đèn bằng sb và lb.

- Thực hiện phép đổi tín hiệu đèn bằng cách đảo bit của tín hiệu đèn hiện tại bằng phép toàn xori với 0xFF.

1. **Mã nguồn**

#include <iregdef.h>

#include <idtcpu.h>

#include <excepthdr.h>

#define PIO\_SETUP2 0xffffea2a

#define SWITCHES 0xbf900000 # Dia chi cua ngan nho, ma ngan nho do tuong ung voi cong tac SWITCH

#define LEDS 0xbf900000 # Dia chi cua ngan nho, ma ngan nho do tuong ung voi den led, trung dia chi SWITCH.

#define BUTTONS 0xbfa00000 # Tuong tu nhu SWITCH

.text

# Interrupt routine. Uses ra, a0, a1, a2, and a3.

# It is also necessary to save v0, v1 and t0-t9

# since they may be used by the printf routine.

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# GENERAL INTERRUPT SERVED ROUTINE for all interrupts

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

.globl introutine

.ent introutine

.set noreorder

.set noat # Not warning if the AT register is used

introutine:

#--------------------------------------------------------

# SAVE the current REG FILE to stack

#--------------------------------------------------------

subu sp, sp, 22\*4 # Allocate space, 18 regs, 4 args

sw AT, 4\*4(sp) # Save the registers on the stack

sw v0, 5\*4(sp)

sw v1, 6\*4(sp)

sw a0, 7\*4(sp)

sw a1, 8\*4(sp)

sw a2, 9\*4(sp)

sw a3, 10\*4(sp)

sw t0, 11\*4(sp)

sw t1, 12\*4(sp)

sw t2, 13\*4(sp)

sw t3, 14\*4(sp)

sw t4, 15\*4(sp)

sw t5, 16\*4(sp)

sw t6, 17\*4(sp)

sw t7, 18\*4(sp)

sw t8, 19\*4(sp)

sw t9, 20\*4(sp)

sw ra, 21\*4(sp)

# Note that 1\*4(sp), 2\*4(sp), and 3\*4(sp) are

# reserved for printf arguments

.set reorder

#--------------------------------------------------------

# Detect the CAUSE of Interrupt, maybe K1, K2, Timer and

# the instruction address in the main program when it happens (to return later).

#--------------------------------------------------------

mfc0 k0, C0\_CAUSE # Retrieve the cause register

mfc0 k1, C0\_EPC # Retrieve the EPC

#--------------------------------------------------------

# Get the I/O port address

# Used to detect K1, K2, timer were pressed

#--------------------------------------------------------

lui s0, 0xbfa0 # Place interrupt I/O port address in s0

#--------------------------------------------------------

# The main function of GENERAL INTERRUPT SERVED ROUTINE

# Print CAUSE, EPC, I/O Port to console

#--------------------------------------------------------

#Step 2 ------------------------------------------------

check\_k1:

addi t4,zero,1

bne t4,t5,check\_k2 #Kiem tra xem co trong vong loop cua k1 hay khong. Neu khong thi khong kiem tra dieu kien 3 lan va check xem co phai dang loop cua k2 khong

nop

addi t4,zero,3

bne t3,t4,k1s #Kiem tra neu k1 chua dem du 3 lan; t3 de luu so lan lap

nop

addi t5,zero,0 # Du 3 lan thi reset bien flag t5

addi t3,zero,0 # Du 3 lan thi reset bien dem

addi t2,t1,0 # Du 3 lan thi reset trang thai den

xori t2,t2,0xFF # reset cho lan dau tien

check\_k2:

addi t4,zero,1

bne t4,t6,step2\_t #Kiem tra xem co trong vong loop cua k1 hay khong. Neu khong thi khong kiem tra dieu kien 3 lan va check xem co phai dang loop cua k2 khong

nop

addi t4,zero,3

bne t3,t4,k2s #Kiem tra neu k1 chua dem du 3 lan; t3 de luu so lan lap

nop

addi t6,zero,0 # Du 3 lan thi reset bien flag t6

addi t3,zero,0 # Du 3 lan thi reset bien dem

addi t2,t1,0 # Du 3 lan thi reset trang thai den

xori t2,t2,0xFF # reset cho lan dau tien

step2\_t:

lbu a3, 0x0(s0) # Read the interrupt I/O port

li t4,0x20 # Vi k1 = 0x01000x0

and t4,a3,t4

bne t4,zero,k1t # Kiem tra xem tin hieu ngat co phai do k1 gay ra khong, co thi jump den k1t

nop #

li t4,0x10 # Vi k2 = 0x01000x0

and t4,a3,t4

bne t4,zero,k2t # Kiem tra xem tin hieu ngat co phai do k2 gay ra khong, co thi jump den k2t

nop #

xori t2,t2,0xFF # Dao gia tri cua den

sb t2, 0(t0) # Hien thi ra den led

nop

j ends # Ket thuc 1 chu ki

nop

k1t:

addi t3,zero,0 # Neu dang trong luc lap k1, bam k1 thi tinh lai tu dau

addi t5,zero,1 # t5 luu gia tri de xac dinh la loop vi k1

k1s:

addi t2,zero,0xF0 #

sb t2, 0(t0) # Hien thi den nhu yeu cau

addi t3,t3,1 # Tang bien dem so chu ki len 1

j ends

nop # Ket thuc 1 chu ki

k2t:

addi t3,zero,0 # Neu dang trong luc lap k2, bam k2 thi tinh lai tu dau

addi t6,zero,1 # t6 luu gia tri de xac dinh la loop vi k2

k2s:

addi t2,zero,0x3C #

sb t2, 0(t0) # Hien thi den nhu yeu cau

addi t3,t3,1 # Tang bien dem so chu ki len 1

j ends

nop

#End of step 2 -----------------------------------------

ends: # Ket thuc 1 chu ki xu li ngat

#--------------------------------------------------------

# Reset the I/O port address to zero after serving interrupt

#--------------------------------------------------------

sb zero,0x0(s0) # Acknowledge interrupt, (resets latch)

#--------------------------------------------------------

# RESTORE the REG FILE from STACK

#--------------------------------------------------------

.set noreorder

lw ra, 21\*4(sp) # Restore the registers from the stack

lw t9, 20\*4(sp)

lw t8, 18\*4(sp)

lw t7, 18\*4(sp)

# lw t6, 17\*4(sp)

# lw t5, 16\*4(sp)

lw t4, 15\*4(sp)

# lw t3, 14\*4(sp)

# lw t2, 13\*4(sp)

lw t1, 12\*4(sp)

lw t0, 11\*4(sp)

lw a3, 10\*4(sp)

lw a2, 9\*4(sp)

lw a1, 8\*4(sp)

lw a0, 7\*4(sp)

lw v1, 6\*4(sp)

lw v0, 5\*4(sp)

lw AT, 4\*4(sp)

addu sp, sp, 22\*4 # Return activation record

#--------------------------------------------------------

# noreorder must be used here to force the

# rfe-instruction to the branch-delay slot

jr k1 # Jump to EPC

rfe # Return from exception

# Restores the status register

.set reorder

.end introutine

# The only purpose of the stub routine below is to call

# the real interrupt routine. It is used because it is

# of fixed size and easy to copy to the interrupt start

# address location.

.ent intstub

.set noreorder

intstub:

j introutine

nop

.set reorder

.end intstub

.globl start # Start of the main program

.ent start

start:

#----------------------------------------------------------

# Enable IO PIN of K1, K2 buttons

#----------------------------------------------------------

lh a0, PIO\_SETUP2 # Enable button port interrupts

andi a0, 0xbfff

sh a0, PIO\_SETUP2

#----------------------------------------------------------

# Reset I/O port address before enable interrupts

#----------------------------------------------------------

lui t0, 0xbfa0 # Place interrupt I/O port address in t0, t0 = 0xbfa0.0000

sb zero,0x0(t0) # Acknowledge interrupt, (resets latch)

#----------------------------------------------------------

# Register the GENERAL INTERRUPT SERVED ROUTINE

#----------------------------------------------------------

la t0, intstub # These instructions copy the stub

la t1, 0x80000080 # routine to address 0x80000080

lw t2, 0(t0) # Read the first instruction in stub

lw t3, 4(t0) # Read the second instruction

sw t2, 0(t1) # Store the first instruction

sw t3, 4(t1) # Store the second instruction

#----------------------------------------------------------

# Set the status register to ENABLE EXPECTED INTERRUPTS

# such as K1, K2, timer and ENABLE THE GENERAL INTERRUPT

#----------------------------------------------------------

mfc0 v0, C0\_SR # Retrieve the status register ,v0 = status\_register

li v1, ~SR\_BEV # Set the BEV bit of the status ,SR\_BEV = 0x00400000 /\* use boot exception vectors \*/

and v0, v0, v1 # register to 0 (first exception vector) ,v0 = status\_register and (not SR\_BEV)

ori v0, v0, 1 # Enable user defined interrupts ,v0 = status\_register and (not SR\_BEV) or SR\_IEC, /\* SR\_IEC= cur interrupt enable, 1 => enable \*/

ori v0, v0,EXT\_INT3 # Enable interrupt 3 (K1, K2, timer) ,v0 = status\_register and (not SR\_BEV) or SR\_IEC or EXT\_INT3

mtc0 v0, C0\_SR # Update the status register ,status\_register = status\_register and (not SR\_BEV) or SR\_IEC or EXT\_INT3

#----------------------------------------------------------

# No-end loop, main program, to demo the effective of interrupt

#----------------------------------------------------------

#Step1 ----------------------------------------------------

step1:

li t0, SWITCHES # Lay dia chi cua SWITCH

lb t1, 0(t0) # Doc trang thai SWITCH va luu vao t1

nop

li t0, LEDS # Lay dia chi cua den Led

sb t1, 0(t0) # Hien thi ra den led

nop

addi t2,t1,0 # Bien temp de hien thi den

addi t3,zero,0 # Bien dem bang 0

addi t5,zero,0 # Flag xac dinh xem co dang trong loop cua k1 khong

addi t6,zero,0 # Flag xac dinh xem co dang trong loop cua k2 khong

#End of step 1 --------------------------------------------

Loop:

nop

nop

nop

nop

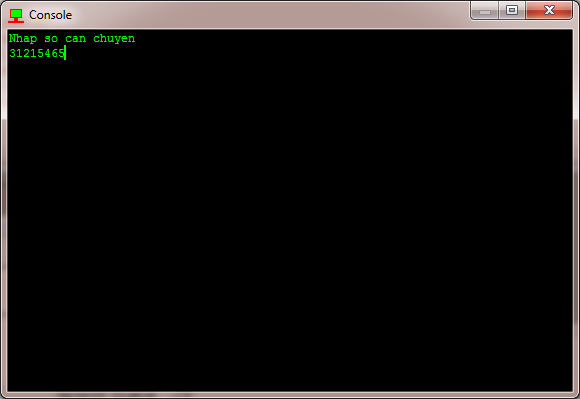
nop

b Loop # Wait for interrupt

.end start

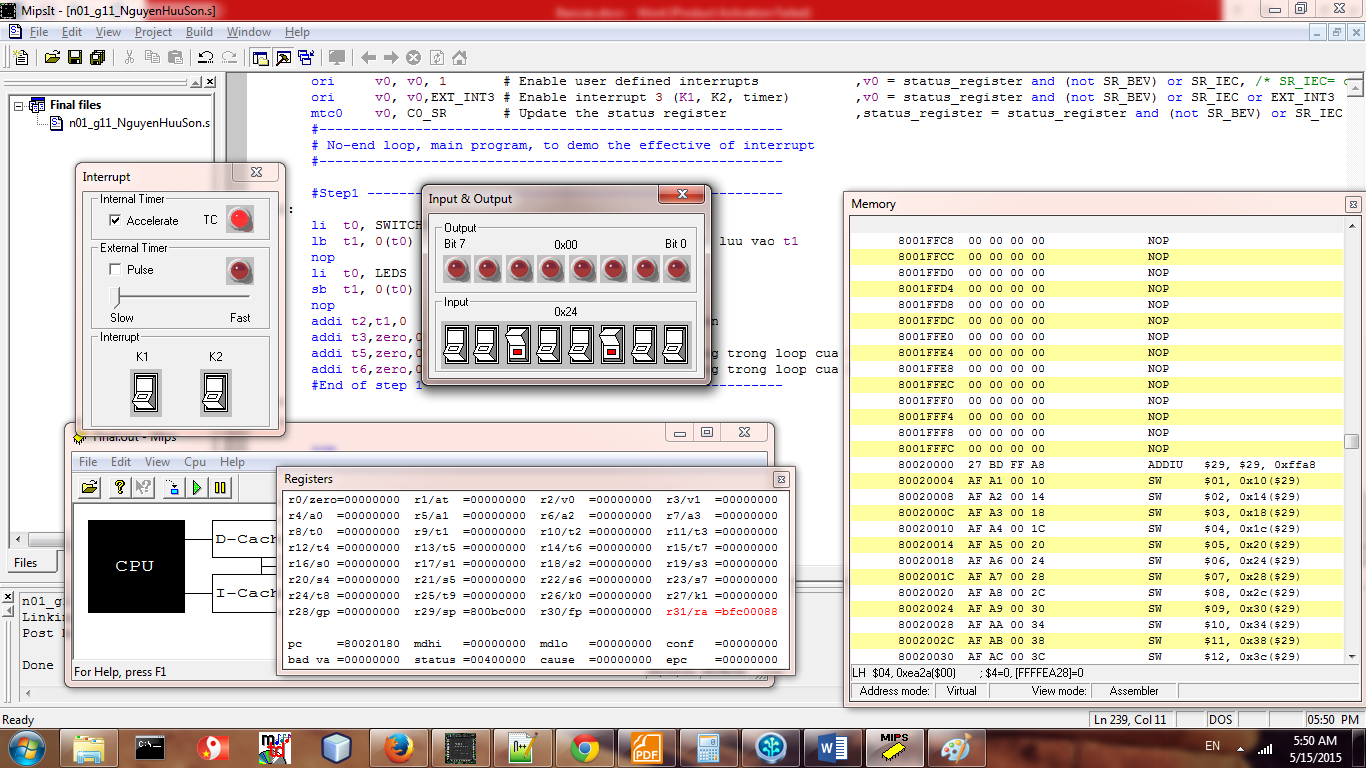
**C. Kết quả chạy mô phỏng**

**I. Nhập dữ liệu**

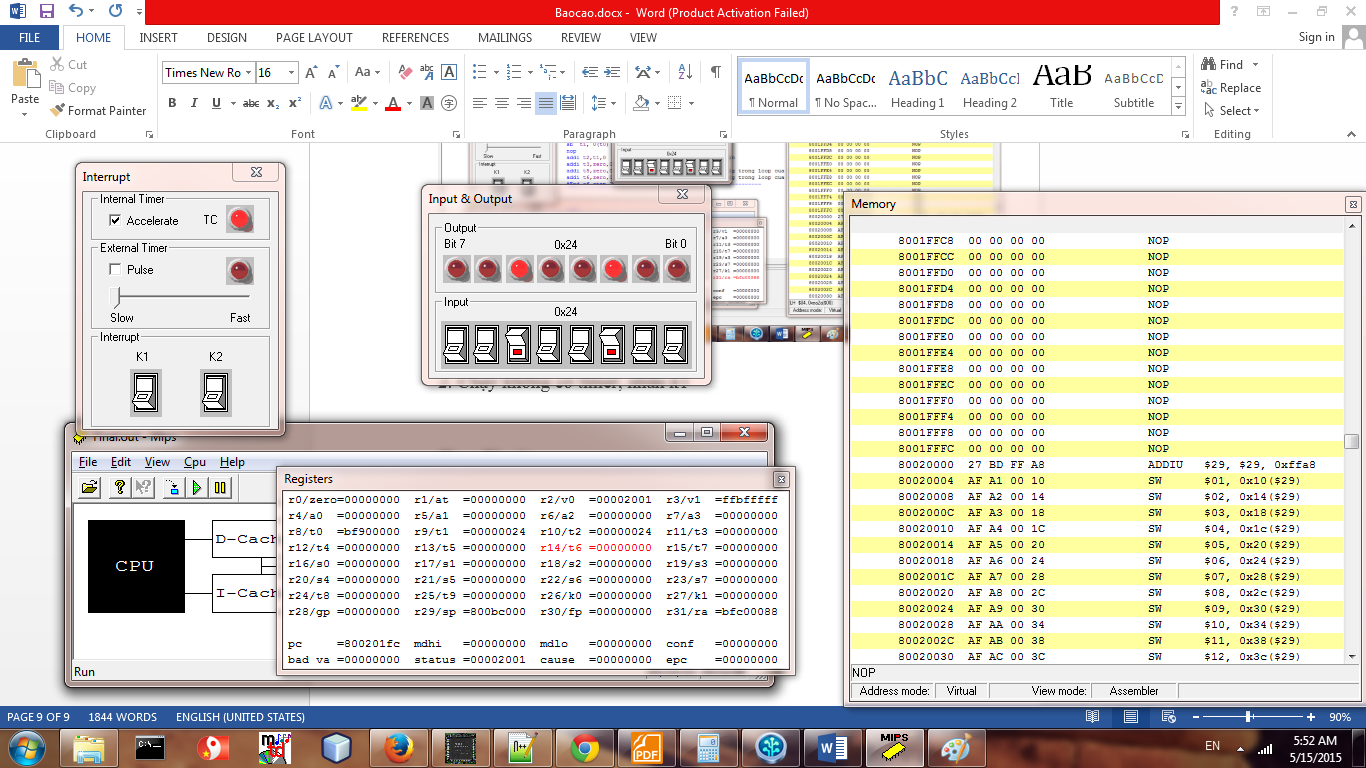


**II. Hiển thị kết quả và lựa chọn thoát**

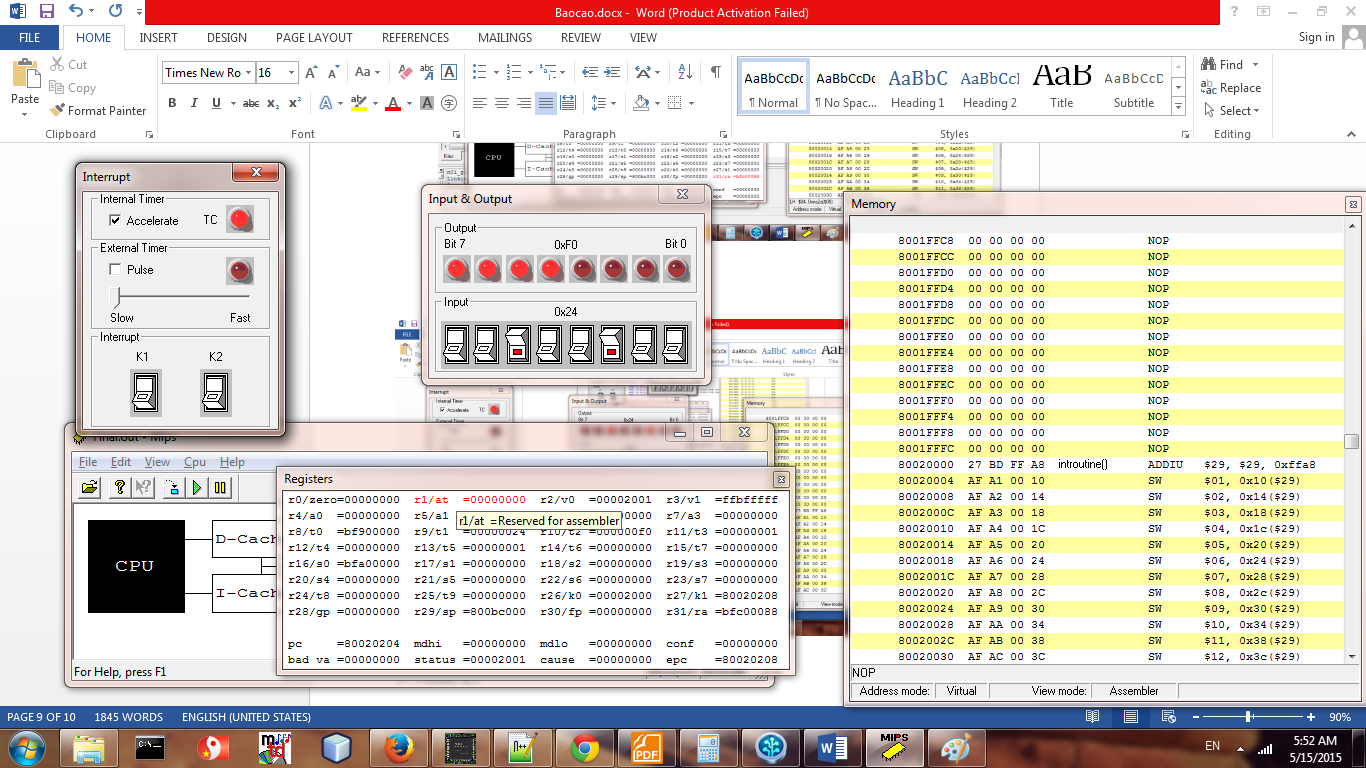
1. Trước khi chạy.



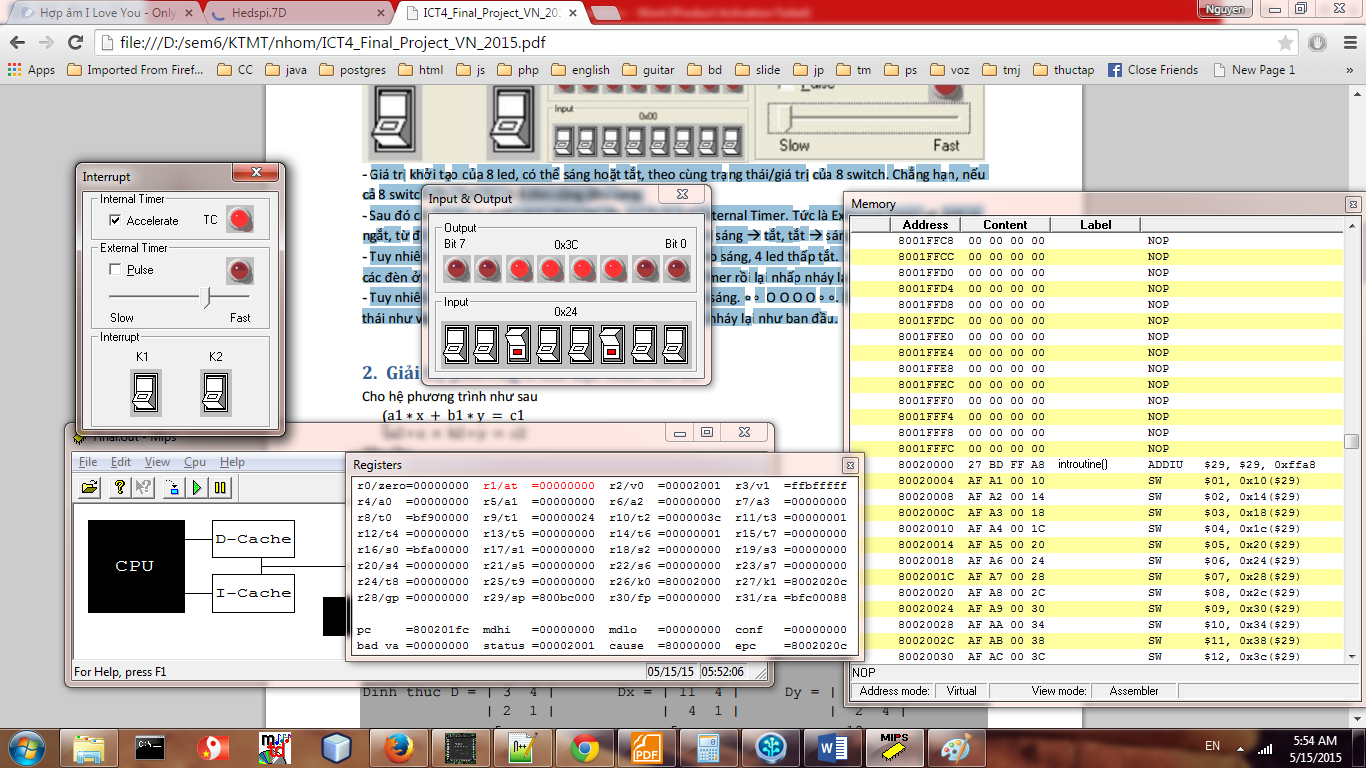
2. Chạy không có timer



3. Nhấn k1



4. Nhấn k2



5. Bật timer, đảo đèn

