Laboratories 4&5 ***AVR Programming***

Instructor: Dr. Lihong Zhang

1 Objectives

With the completion of these labs, you should be able to:

- Write subroutines
- Use the debugging features of the AVR Studio to locate any bugs in your code
- Use assembly data directives to allocate memory for your data

2 Introduction

In these labs, you will write a game called Rat-bashing, in which the objective is to quickly respond to a lighted LED by pressing the appropriate push-button switch. You will implement the key functions of the program as subroutines that you will call in your main program. Carefully read all of the specifications before starting your design – don't start coding until you have a good idea of exactly what is required. Writing brilliant code is great, but of little use if it does not solve the given problem.

As you write your program, take care to document your code using comments. Include a comment block in the beginning to include information about your program, such as its title, your names, the date, a description of the program, and brief instructions. Each key assembly code instruction should have a brief comment that states its purpose. Use the ".EQU" command near the beginning of your .ASM file to establish sensible labels, which can self-document your code. Use ".DEF" for defining registers as variables, for instance:

```
.EQU  ON = 0X00
.EQU  OFF = 0XFF
.EQU  TRUE = 1
.DEF  LED = R16
.DEF  SWITCH = R17
```

For the details and examples of those directives, you can refer to the Assembler Manual by click Help -> Assembler Help within AVR Studio.

3 Rat Bashing

3.1 Problem Statement

Write an assembly language program for the AVR microcontroller that simulates a carnival rat bashing game. The purpose of the real game is to use a play hammer to smash a wooden rat as it pops up through one of several holes. The

rat only remains up for a moment before it goes back down. If the player hits enough rats, he or she wins a prize.

3.2 Specification

In this lab, the STK600 built-in LEDs and push-button switches will be used as display facility and hammer, respectively. **It is defined that PORTB is connected to SWITCHES and PORTD is connected to LEDS.**
PORTB ⇔ STK600 built-in SWITCHES
PORTD ⇔ STK600 built-in LEDS

Therefore, you have to use a 10-wire cable to connect PORTB header and SWITCHES header, and another 10-wire cable to connect PORTD header and LEDS header.

When the player runs the program at the STK600 board, the program must indicate it is about to start by blinking all the LEDs on and off three times. It must then wait for about 1 second with no LEDs lit.

Next, the program must show a single LED representing the rat to be bashed. It must then wait (for about a second) for the player to hit the rat by pressing the corresponding push-button switch. Register a hit if the player presses only the correct switch - if another switch is also pressed; consider it to be a miss. When a hit has been registered, the lit LED should quickly blink (off-on) 3 times. On a miss, do nothing special. Follow either case with approximately a second of all lights extinguished.

Repeat this for 8 rounds, with a variety of different LEDs lit. The pattern should not be predictable within a single game, but does not have to vary each time the game is played.

When done, display on the LEDs the rats that were hit. The LED number should represent the round. For example, if the player hit the rat on rounds 1, 2, 5, 7, and 8, then lights LED0, LED1, LED4, LED6, and LED7 should light. If all 8 rats were hit, show a fancy display – something worthy of the feat that the player has accomplished.

The program must include at least the following three subroutines; you have some flexibility with your implementation, and may include more functionality than listed here:

DisplayRat - displays the next rat on the LEDs
HammerWait - waits for a limited time for the player to press a switch
ShowScore - displays the rats that were hit during the game

3.3 Implementation Notes & Suggestions

Your code should consist of a main block of code, in which you call various subroutines. The main code should first initialize any registers or port configuration, and then enter a loop. After looping, finish by displaying the score or fancy display on the LEDs.

If you wish to store all 8 LEDs in memory using ".DB", and then manually set them at assembly time, that is fine. On the other hand, if you can determine a way to randomly generate the next rat location, please do so!

When you design your subroutines, you may require that a certain register or memory location be set before the main code calls the subroutine. For example, your DisplayRat routine might require register Z to point to the memory location where the next rat is stored.

3.4 Debugging

Use the debugging skills you acquired in Lab 2 to hunt down any anomalies in your program. Do **not** call over a TA as soon as you realize that the program is not working as you expect.

4 Submission

When you have finished your program, demonstrate it to a TA. When the TA is satisfied that it is working correctly, print off your program and she or he will sign it for you. If you submit a version of your program that does not meet the specifications, you will not be able to receive full marks on the lab.

As your report, **submit a short document** describing your program. Include a brief overview of your program (what it is and how to use it). Explain any major design decisions. List all of subroutines, giving their names, brief descriptions, any registers or memory locations that must be set prior to calling, and any registers that will be set after the return instruction. Describe any difficulties you faced and the way you solved it.