CSSE1000/CSSE7035 - Prac 6

AVR Assembly Language Programming

Goal

- Gain some experience with AVR assembly language programming.
- Using assembly language you should understand:
  - How the I/O ports of AT90S8515 are used as inputs and outputs.
  - The polling method for sensing inputs.

See the following topics:

- Preparation
- Procedure
- Equipment
- References

Back to Lab Experiment Index

Preparation

Each student has a slightly different preparation task - based on your login. You must login to this page using your own student username.

You are currently logged in as: s4209435

Make sure you have completed and understand prac 5 and the AVR Studio tutorial before you try to prepare for this prac. This prac assumes that you understand this material.

Task 1: Accumulator

Write **and simulate** (using AVR Studio) an Atmel AVR program that has I/O ports connected to an 8 bit input "S" (which will be 8 toggle switches), an 8 bit output "L" (which will be connected to 8 LEDs) and a push button input "P". When reset (e.g. on program/device start-up), the L output is set to 0. Each time P is pressed the value on the S input is added to L.

For example, if after a reset we set S to 64 (01000000 binary) and press P, L should update to 64 (01000000 binary). If we then change S to 31 (00011111 binary) and press P again, L should update to 95 (64 + 31 = 01011111 binary). If we then change S to 21 and press P again, L will update to 116 (95 + 21).

- Use Port A on the AVR for S (toggle switches)
- Use Port B for L (LEDs)
- Use Port C pin 0 for P (the push button)

HINT: Use the **cpi** and **breq** instructions to continually wait for and detect the button being pressed. Use the **andi** instruction to mask out the push button input from the 8 port C pins. Check out the instructions in the instruction reference below. Note that you can't use instructions like **cpi** and **andi** directly on I/O registers - they only work on general purpose registers. To copy data from I/O registers to general purpose registers you will need to use the **in** instruction; to copy data from a general purpose register to an I/O register you will need to use the **out** instruction. Remember also that to input data from the general I/O ports you need to copy data from the PIN registers (e.g. PINA, PINB etc); to output data you copy it to the PORT registers (PORTA, PORTB etc). You also have to set the data direction register bits appropriately.

Task 2: Combination Lock

Write an Atmel AVR program to implement the combination-lock design task from prac 4:

As before, The combination lock system has

- a 2 digit 7-segment hex display for showing the input combination
- 4 toggle switches and a push button to enter the code
- one LED indicating that the lock is open,
- another LED indicating that it is closed
- a reset button (in this case the AVR board reset button).

It operates as follows: After a reset the display is set to "00" and the lock is closed. When the push button is pressed, the value on the 4 toggle switches is saved and the left hand hex digit display is set to that value. When the same button is pressed again, the second digit entered (using the same 4 toggle switches) is saved and is displayed on the left hand hex digit display. The digit that was displayed there moves to the right hand hex digit display. Only the last two digits entered are remembered. If the two digit code displayed is correct, the lock is opened with the "open" LED turned on. (Otherwise, the "close" LED remains lit.)

The correct code for the combination lock is: 83

- Use Port A pins 3-0 for the left hand display digit.
- Use Port B pins 3-0 for the right hand display digit.
- Use Port C pins 3-0 for the toggle switch inputs.
- Use Port C pin 4 for the push button.
- Use Port D pin 0 for the *open* LED
- Use Port D pin 1 for the *closed* LED

Instruction reference

You may find the following instructions useful for either or both tasks.

Instruction	Usage	Example	Meaning
cpi	cpi reg, value	cpi r18, $FF	Compares the given value to the contents of the given general purpose register (reg) and sets the status register bits appropriately. (Only works on registers r16 to r31)
breq	breq label	breq my_label	Jump to the given location if compared values are equal. I.E. when used immediately after a compare instruction (such as cpi), if the two compared values are equal then execution jumps to the given program location
add	add reg1, reg2	add r11,r21	Add the contents of the second general purpose register to the first. (reg1 <- reg1 + reg2)
andi	andi reg, value	andi r17, $01	ANDs the contents of the given general purpose register with a constant value. **Very useful for masking bits.** (Only works on registers r16 to r31)
brne	brne label	breq my_label	Like breq, but instead branch if the compared values are not equal
in	in reg, ioreg	in r17,PINA	Load the contents of an I/O register (ioreg) into a general purpose register (reg)
ldi	ldi reg, value	ldi r18, 128	Loads the given value into the given general purpose register (reg). (Only works on register r16 to r31)
out	out ioreg, reg	out PORTA, r18	Store the contents of a general purpose register (reg) to an I/O register (ioreg)
rjmp	rjmp label	rjmp mainloop	Jump to the given program location.

Procedure

1. Simulate your and your partner's task 1 (accumulator) code and discuss your results with your

partner. Note any differences between how your two programs operate. Make notes in your workbook.
2. Download one of your working accumulator designs to the board. Wire it up, test it, and document your experimental results in your workbook.
3. Simulate your and your partner's task 2 (combination lock) code and discuss your results with your partner. Note any differences between how your two programs operate. Make notes in your workbook.
4. Download one of your working combination lock designs to the board. Wire it up to the logic workstation as specified. Test it, and document your experimental results in your workbook.
5. Demonstrate Part 4 and show your completed workbook to the tutor.
6. Tutor Task - the tutor will ask you to modify your combination lock design to meet a slightly different specification.

Back to Contents

Assessment

This practical is marked out of 4 and worth 2% of your mark for CSSE1000/CSSE7035:

- Preparation - AVR assembly language code written (using supplied values for your login to this page) - ***1 Mark***
- Demonstration - Required circuit and program demonstrated. - ***1 Mark***
- Documentation - All testing results are documented in workbook - ***1 Mark***
- Tutor Task - Tutor task completed and documented in workbook - ***1 Mark***

Back to Contents

Equipment

- Computer
- 12V Power Supply
- AVR8515 Project Board with STK200 Download Cable
- Logic Workstation
- Hookup Wire

Back to Contents

References

- Atmel AVR Resources
- AVR Tutorial
- PonyProg Tutorial

Back to Contents