Lab 6

Introduction to Assembly Language Programming of AVR Microcontrollers

Objectives

- Implementing functions, loops, and conditional execution in AVR Assembly language.
-

Software

- Microchip Studio (Version 7)
- Proteus ISIS (Version 8.4)
- AVRDUDE

Theory

This lab would introduce students to the ATmega328P assembly language comprising 132 instructions. There are various advantages of using assembly, including, but not limited to:

- Program written in assembly is faster to execute
- The execution time can be measured precisely
- It is possible to generate very specific delays
- Code with a small program size can be generated
- Programmers can control the controller directly without abstractions
- Programmers can optimize the code for latency

Whenever a program for ATmega328P is compiled, a .hex file is generated that comprises the machine coded version of the program. Assembly language is an English type representation of machine code. Every line machine code can be represented by a corresponding line of assembly code, and vice versa.

An assembly language program written by a programmer can be "assembled" into machine code. Once the machine code (.hex file) is loaded into the program memory of the microcontroller, the ALU starts executing each instruction 1 at a time. The ATmega328P operates in a two-stage pipeline.

In every instruction cycle, every instruction is fetched, then it is decoded (to ascertain the "instruction" and the operands), executed and stored back.

All assembly language instructions are 1 of 7 kinds:

1. Mathematical instructions

2. Logical instructions
3. Bit manipulation instructions
4. Branching instructions
5. Data transfer instructions
6. I/O instructions
7. Configuration and control of MCU components (Timers, ADCs, serial comms etc.)

Invariably, almost all instructions involve movement of data between general purpose registers (GPRs) and other memory components. This is because all mathematical, logical, and branching instructions work with general purpose registers (GPRs) only. The types of movements are illustrated in figure 6.1. and the instructions summarized in table 6.1.

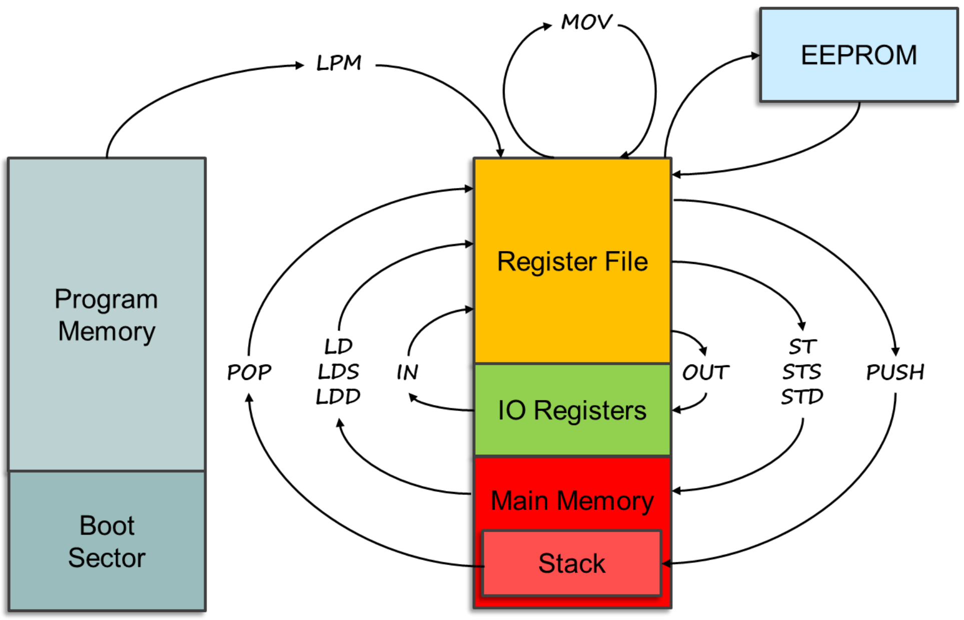

Figure 6.1 – A summary of all possible data movements and their corresponding assembly language instruction

Instruction	Meaning	Role	Example
mov	Move	Move the data from one GPR to another GPR	MOV R16, R17
st/sts/std	Store	Move data from the GPR to main memory (RAM)	ST X, R16
ld/lds/ldd	Load	Move data from main memory in the GPR	LD R16, X
in	In	Move data (in) from chip pins to GPR	IN R16, PINB
out	Out	Move data (out) from GPR to pins	OUT PORTB, R16
push	Push	Push data from a GPR to the top of the system stack	PUSH R16
pop	Pop	Pull data from the top of the system stack into a GPR	PULL R16
lpm	Load (from) Program Memory	Load data from the program memory into a GPR	

Table 6.1 – Data Movement Instructions

Pre-Lab Tasks

Task-1

Consider the following C code. Your task is to write an AVR Assembly program with the same functionality.

C-Code

```
include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>

int main()
{

     DDRB = 0xFF;
     while(1){
          PORTB |= (1<<PB3);
          _delay_ms(100);
          PORTB &= ~(1<<PB3);
          _delay_ms(100);
     }
}
```

In-Lab Tasks

Task-1

Generate a square wave signal of frequency 1 KHz and 40% duty cycle on a digital I/O pin of ATmega328P.

Task-2

Generate a square wave of 50% duty cycle on PD6. The user should be able to choose the frequency (in kHz) of the signal by the input from the 4-DIP switch attached to PortD (1 – 16 kHz).

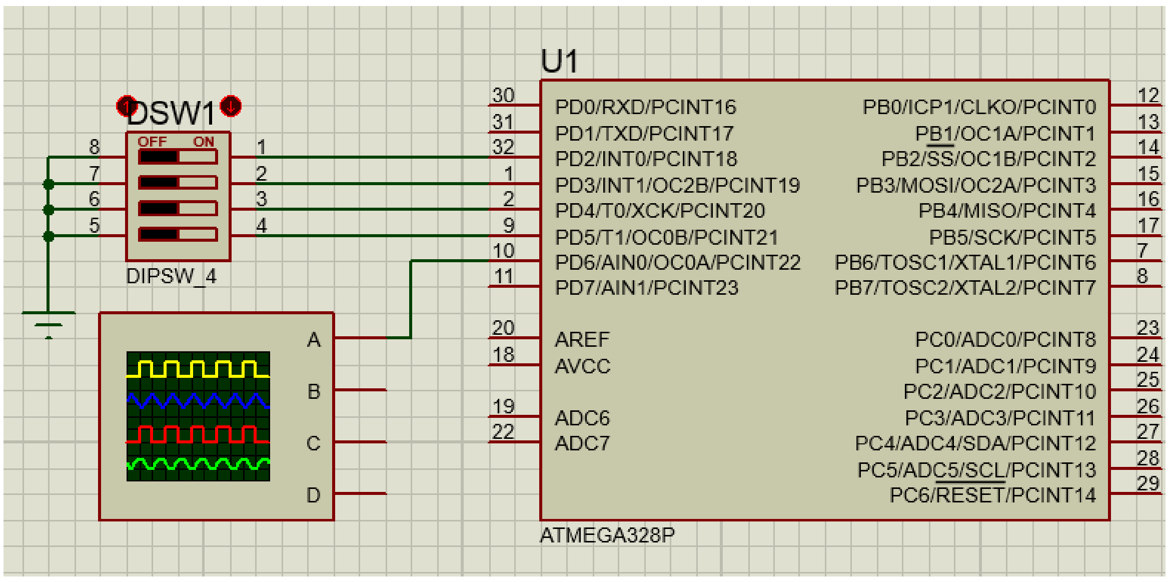

Figure 6.2 – Circuit diagram for Task 2

Task-3

Download the code from Task 2 the MCU and test the output on an Oscilloscope.

Post Lab

Task-1

Write a report explaining the problems faced in implementing the in-lab tasks and provide their solutions.

Critical Analysis / Conclusion (To be filled in by the student)

Lab Assessment (To be filled by the lab-instructor)		
Pre-Lab	/5	/25
In-Lab	/5	
Results	/5	
Viva	/5	
Critical Analysis	/5	
Comments:		
	Instructor Name	Instructor Signature