Assembly Programming for the ATMega328 CPU on the Arduino Uno

Reference:

https://dumblebots.com/2022/07/31/programming-arduino-and-avr-microcontrollers-using-the-assembly-language/

https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoISP/

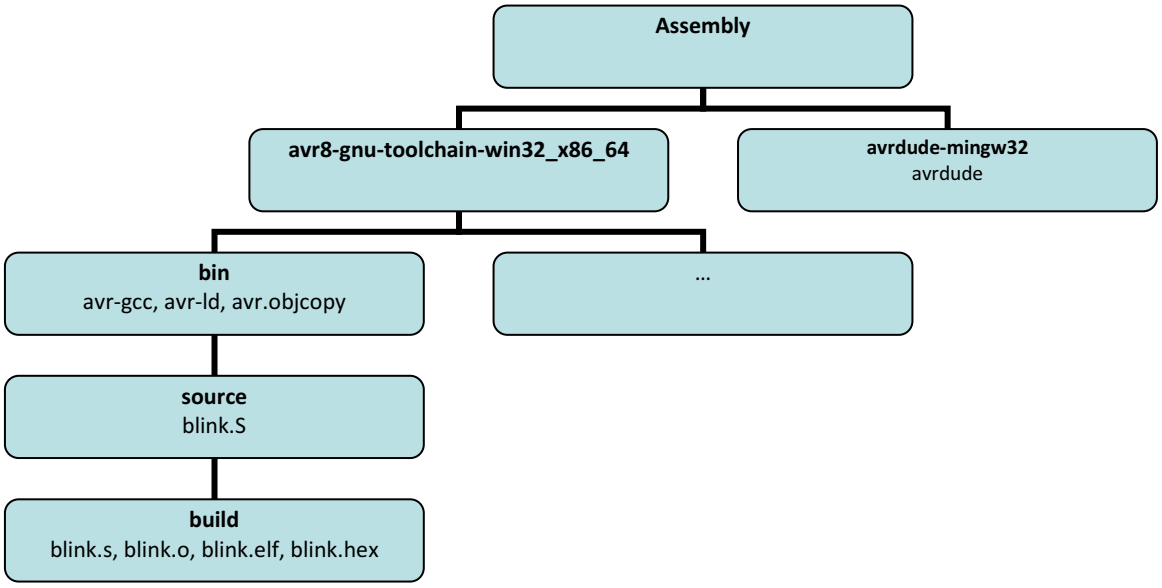
This document is a summary of the steps extracted from the above two references needed to write, compile and upload assembly code to an ATMega328 CPU on an Arduino Uno board.

This document assumes you know how to work in the command window under Windows or Linux. All the AVR Toolchain commands will be issued inside the command window.

1. ATMega328 CPU

2. Create this directory structure

Directory names are bolded.



3. Download the AVR Toolchain commands

All the commands are executed from the command window. They must be issued from the same directory that they are located in since the system PATH variable doesn't have the paths to these directories. Alternatively, you can add these paths to the PATH variable so that these commands can be executed from any directory.

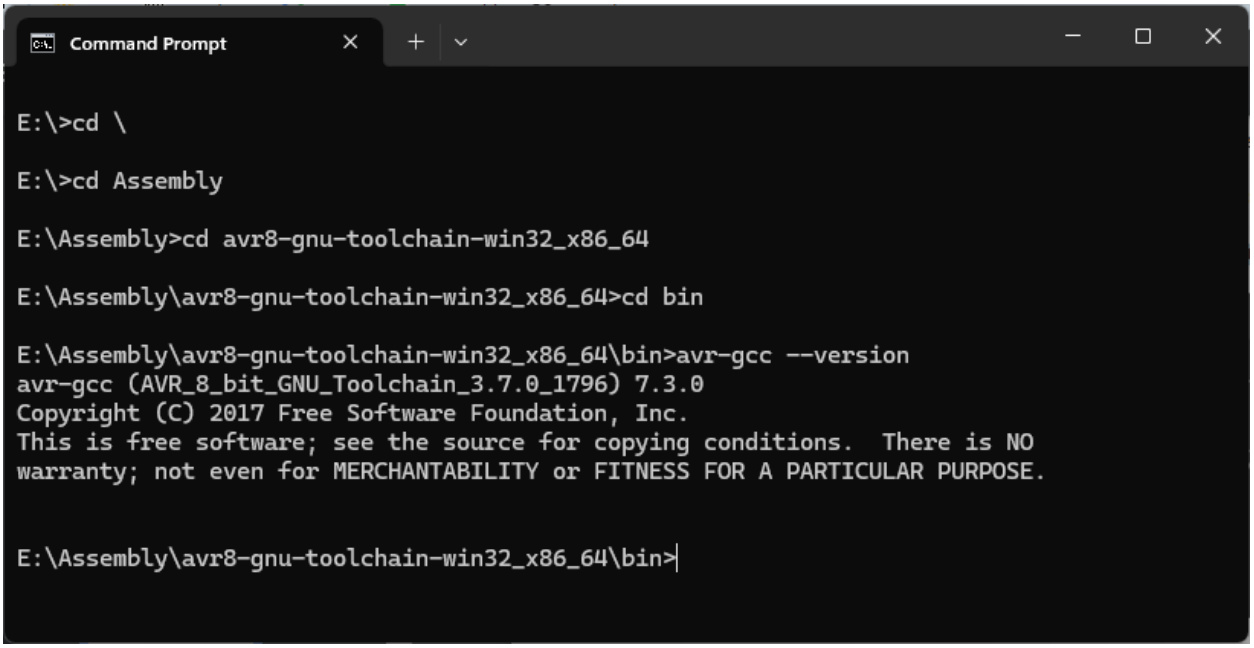
- Download the GCC compiler **AVR 8-Bit Toolchain** (for Windows, Linux, or OSX) from

  https://www.microchip.com/en-us/tools-resources/develop/microchip-studio/gcc-compilers

  Copy the folder **avr8-gnu-toolchain-win32_x86_64** in the zip file to inside the **Assembly** folder.

  Open a command window. Go to the **bin** folder inside this directory.

  To test your installation, enter the command **avr-gcc –version**



- Download the programmer (uploader) **avrdude-6.4-mingw32.zip** (or the latest version) from

  https://download.savannah.gnu.org/releases/avrdude/

  Copy the contents of the zip file into a directory named **avrdude-mingw32**.

  Go to this directory.

  To test your installation, enter the command **avrdude**.

```
Command Prompt

E:\Assembly\avr8-gnu-toolchain-win32_x86_64\bin>cd \

E:\>cd Assembly

E:\Assembly>cd avrdude-mingw32

E:\Assembly\avrdude-mingw32>avrdude
Usage: avrdude [options]
Options:
  -p <partno>                Required. Specify AVR device.
  -b <baudrate>              Override RS-232 baud rate.
  -B <bitclock>              Specify JTAG/STK500v2 bit clock period (us).
  -C <config-file>           Specify location of configuration file.
  -c <programmer>            Specify programmer type.
  -D                         Disable auto erase for flash memory
  -i <delay>                 ISP Clock Delay [in microseconds]
  -P <port>                  Specify connection port.
  -F                         Override invalid signature check.
  -e                         Perform a chip erase.
  -O                         Perform RC oscillator calibration (see AVR053).
  -U <memtype>:r|w|v:<filename>[:format]
                             Memory operation specification.
                             Multiple -U options are allowed, each request
                             is performed in the order specified.
  -n                         Do not write anything to the device.
  -V                         Do not verify.
  -u                         Disable safemode, default when running from a script.
  -s                         Silent safemode operation, will not ask you if
                             fuses should be changed back.
  -t                         Enter terminal mode.
  -E <exitspec>[,<exitspec>] List programmer exit specifications.
  -x <extended_param>        Pass <extended_param> to programmer.
  -v                         Verbose output. -v -v for more.
  -q                         Quell progress output. -q -q for less.
  -l logfile                 Use logfile rather than stderr for diagnostics.
  -?                         Display this usage.

avrdude version 6.4, URL: <http://savannah.nongnu.org/projects/avrdude/>

E:\Assembly\avrdude-mingw32>
```

4. Program an Arduino to use as an In-System Programmer (ISP) for programming the ATMega328 CPU

1. Upload the ArduinoISP sketch to your Arduino Uno by selecting from the Arduino IDE menu

   File / Examples / 11.ArduinoISP

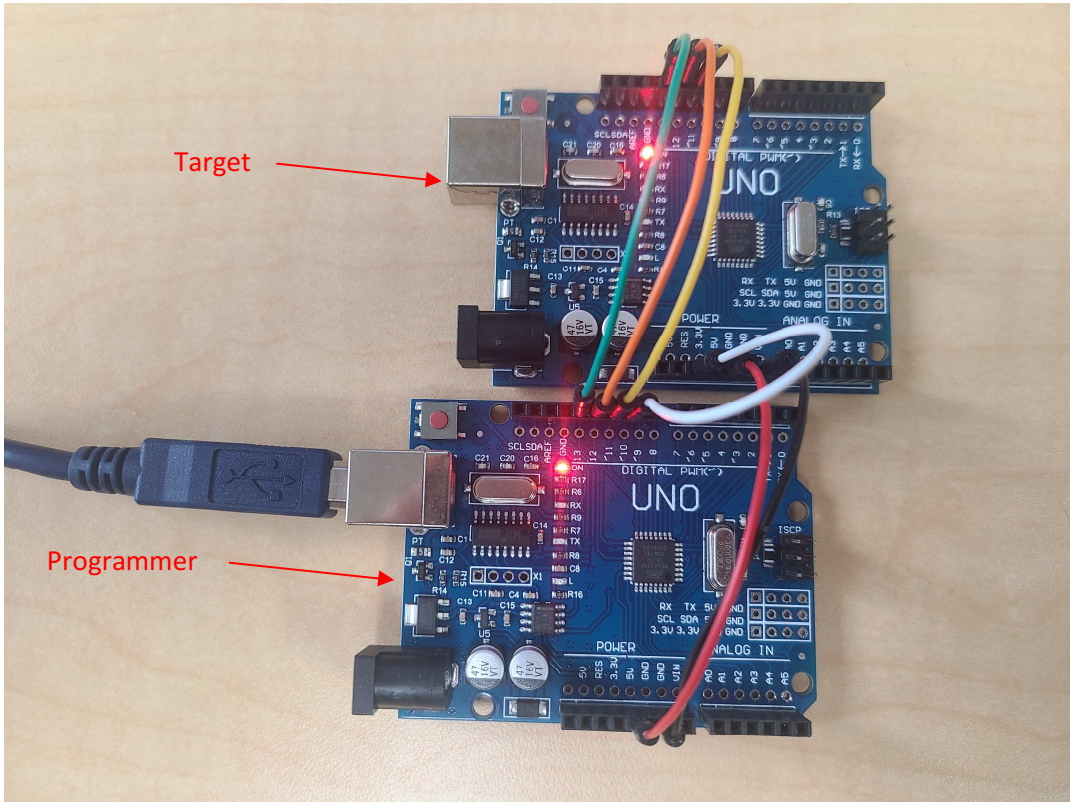
   and click on Upload. Once uploaded your Arduino will be programmed to be used as an ISP.

5. Connections for Programming the ATMega328 CPU

The Uno board with the ArduinoISP sketch (from Step 4) will be referred to as the Programmer.

A second Uno board where you want to upload and run the assembly code on the ATMega328 CPU will be referred to as the Target.

Programmer	**Target**
13 (SCK)	13 (SCK)
12 (MISO)	12 (MISO)
11 (MOSI)	11 (MOSI)
10	Reset
Gnd	Gnd
5V	5V



Plug in the Programmer board and find the COM port that it is connected to.

6. Test Program

Copy the following blink program to a file named **blink.S**. Note that the extension must be capital "S". Save this file inside the **source** directory.

```
; blink program

#include <avr/io.h>

.section .data

.section .bss

.section .text
        .org 0x00
        LDI     R16, (1<<PB5)
        LDI     R17, (1<<PB5)
        OUT     _SFR_IO_ADDR (DDRB), R16

LOOP:       OUT     _SFR_IO_ADDR (PORTB), R16
        RCALL   DELAY_1S
        EOR     R16, R17
        RJMP    LOOP

DELAY_1S:

        LDI     R20, 64         ; about 1 second delay
;        LDI    R20, 20         ; about 0.5 second delay
DELAY1:     LDI     R21, 250
DELAY2:     LDI     R22, 250
DELAY3:     DEC     R22
        NOP
        BRNE    DELAY3

        DEC     r21
        BRNE    DELAY2

        DEC     r20
        BRNE    DELAY1

        RET
```

7. Compile the program

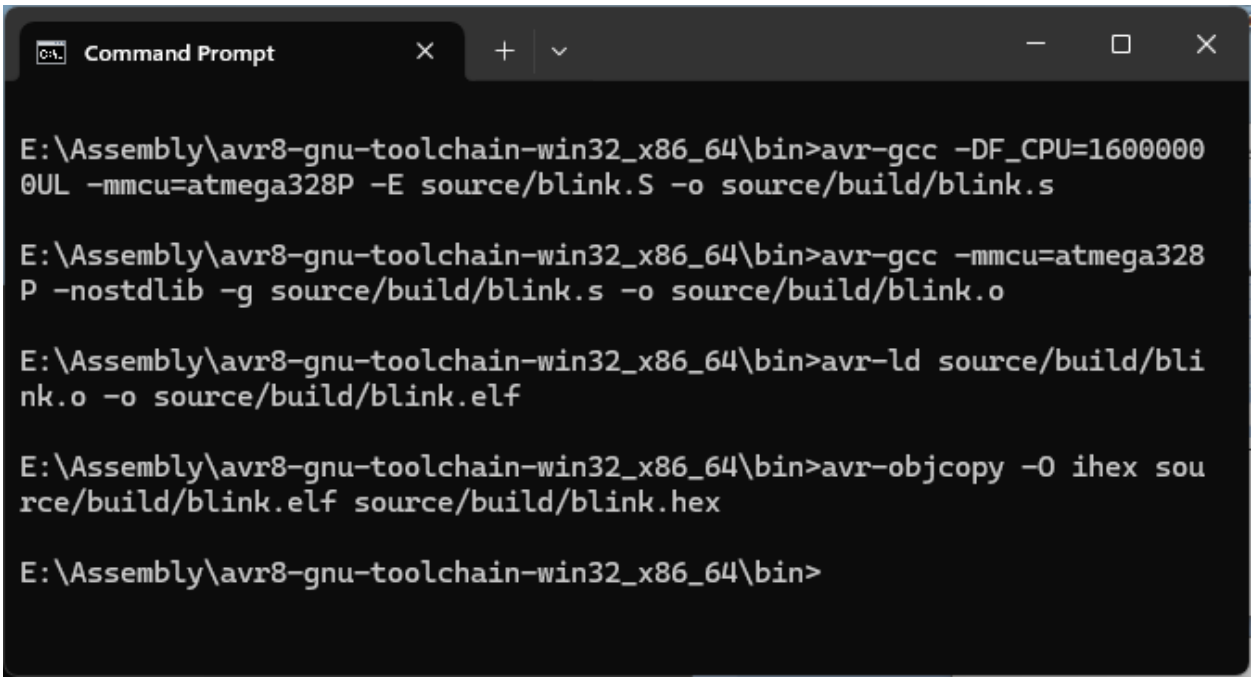
Navigate to the **bin** directory.
blink.S is the source assembly file located under the **source** directory inside **bin**.
Note that the extension for the source file must be "S".
The source file with the extension "S" is used in the first command below.
It must be this extension "S" otherwise it will not work.
Issue the following four commands from inside the **bin** directory.

avr-gcc -DF_CPU=16000000UL -mmcu=atmega328p -E source/blink.S -o source/build/blink.s

avr-gcc -mmcu=atmega328p -nostdlib -g -c source/build/blink.s -o source/build/blink.o

avr-ld source/build/blink.o -o source/build/blink.elf

avr-objcopy -O ihex source/build/blink.elf source/build/blink.hex



You should now have four files in the **build** directory, **blink.s**, **blink.o**, **blink.elf** and **blink.hex**.

8. Upload program to the Target ATMega328 CPU

1. Copy the **blink.hex** file into the **avrdude-mingw32** directory.

2. In the command window, navigate to the **avrdude-mingw32** directory.

3. Execute the command. Replace the COM port number with your port number.

   avrdude -v -p atmega328p -c arduino -P COM6 -b 19200 -U flash:w:blink.hex:i

```
Command Prompt

E:\Assembly\avr8-gnu-toolchain-win32_x86_64\bin>cd ..

E:\Assembly\avr8-gnu-toolchain-win32_x86_64>cd ..

E:\Assembly>cd avrdude-mingw32

E:\Assembly\avrdude-mingw32>avrdude -v -p atmega328p -c arduino -P COM6
 -b 19200 -U flash:w:blink.hex:i

avrdude: Version 6.4
         Copyright (c) Brian Dean, http://www.bdmicro.com/
         Copyright (c) Joerg Wunsch

         System wide configuration file is "E:\Assembly\avrdude-mingw32
\avrdude.conf"

         Using Port                    : COM6
         Using Programmer              : arduino
         Overriding Baud Rate          : 19200
         AVR Part                      : ATmega328P
         Chip Erase delay              : 9000 us
         PAGEL                         : PD7
         BS2                           : PC2
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
```

```
Command Prompt

avrdude: safemode: efuse reads as FD
avrdude: NOTE: "flash" memory has been specified, an erase cycle will b
e performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "blink.hex"
avrdude: writing flash (36 bytes):

Writing | ################################################## | 100% 0.1
4s

avrdude: 36 bytes of flash written
avrdude: verifying flash memory against blink.hex:
avrdude: load data flash data from input file blink.hex:
avrdude: input file blink.hex contains 36 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.0
8s

avrdude: verifying ...
avrdude: 36 bytes of flash verified

avrdude: safemode: lfuse reads as FF
avrdude: safemode: hfuse reads as DE
avrdude: safemode: efuse reads as FD
avrdude: safemode: Fuses OK (E:FD, H:DE, L:FF)

avrdude done.  Thank you.


E:\Assembly\avrdude-mingw32>
```

9. Test the program

1. You should see the blink program running on the target board with the led blinking at about 1 Hz.

2. Modify the code in step 6 to decrease the blink speed to about 0.5 Hz. There is a comment in the code regarding this.

3. Repeat steps 7 to 8 to compile and upload the modified code. You should see the led blinking at about 0.5 Hz.