1 GCC

- GNU Compiler Collectin - compiler system.
- Supports various language, processor and host operating system.
- AVR GCC - Reffering to GCC targeting AVR.
- AVR GCC translates high-level langues to assembly.
- AVR GCC three language - C, C++, Ada.

2 GNU Binutils

Source : Tool Chain overview

- Binary Utilites - contains the GNU assembler(gas), GNU linker(ld), etc.
- avr-as - The assembler.
- avr-ld - The linker.
- avr-objcopy - Copy and translate object files to different format.
- avr-objdump - Display information from object file including disassembly.
- avr-size - List section sizes and total sizes.
- avr-nm - List symbols from objects files.
- avr-strings - List printable strings from files.
- avr-readelf - Display contents of ELF files.
- avr-addr2line - Convert addresses to file and line.

3 avr-lib

Open source standard C Libary - standard C Libary and Libary function specifit to AVR.

4 Compiler Options

-On – for optimization level; n indicates the optimization level; 0 being the default no optimization;

- *-O0* – reduces compilation time and this is default
- *-O* and *-O1* – the compiler tries to reduce code size and execution time, without performing any optimizations that take a great deal of compilation time.
- *-O2* – Optimize even more than *-O1*
- *-O3* – Optimize even more than *-O2*
- *-Os* – Optimize for size and enables all *-O2* optimization but not expected to increase code size
- *-Ofast* – enables all *-O3* optimzations and disregarads strict standard compliance
- *-Og* – Optimize for debugging experience

5 Compilation

- Will create the .obj object binary files.
- Use *avr-gcc* along with following options
  - Optimization option *-On* - use *-Os* generally.
  - Warning option *-Wall* - enables all the warning
  - Debug option *-g* - Produce debugging information.
  - MCU option *-mmcu* - the actual MCU - Supported MCU

  - C file option *-c* - the actual c file.
  - Output file name *-o* - Output file name
- To see the object binary file use *avr-objdump -S fileName.o*

```
avr-gcc -Os -Wall -g -mmcu=atmega8 -c hello.c -o hello.o
```

6 Linking

- Link the bianary object file to binary elf file.
- Use *avr-gcc* along with following options
  - Optimization option *-On* - use *-Os* generally.
  - Warning option *-Wall* - enables all the warning
  - Debug option *-g* - Produce debugging information.
  - MCU option *-mmcu* - the actual MCU - Supported MCU
  - .obj file option - the actual .obj file.
  - Output file name *-o* - Output file name
- To see the object binary file use *avr-objdump -S fileName.elf*

```
avr-gcc -Os -Wall -g -mmcu=atmega8 hello.o -o hello.elf
```

7 Generating the hex file

- The Intel hex file is what we program into procesosr.
- Use *avr-objcopy* along with following options
  - section Option *-j* – which sections to copy - generally .text and .data section
  - Output format option *-O* - what Output format should be used - eg) ihex
  - The input .elf file
  - The output .hex file

```
avr-objcopy -j .text -j .data -O ihex hello.elf hello.hex
```

8 AVRDUDE

8.1 Introduction

- AVRDUDE - AVR Downloader UploDEr is a program for downloading and uploading the on-chip memories of Atmel's AVR microcontroller.
- Can program Flash, EEPROM, fuse ,lock bits and signature bytes.
- Can read or write all chip memory types mentioned above.
- Supports varieous programmers from STK500, AVRISP, mkII, JTAG ICE, PPI, serial bit-bang adapters, etc.
- The STK500, JTAG ICE, etc uses serial port to communicate.
- The JTAGICE, AVRISP, USBasp, USBtinyISP uses USB using *libusb*.

8.2 Command Line Options

- *-p partno* – the mandatory option which specifies the MCU.
- *-b baudrate* – Specify the Baudrate.
- *-c programmer-id* – Specify the pgorammer used. eg)arduino, avrisp, avrisp2, avrispmkII, avrispv2, jtag1, stk500, stk500v1, stk500v2, usbasp, usbtiny, etc.
- *-C config-file* – Configuration data file.
- *-e* – Causes a chip erase of FLash ROM, EEPROM to 0xff and clears all lock bits.
- *-F* – Override device signature check.
- *-P port* – Specifty the port to be used.
- *-u* – Used if you want to write fuse bits - this cuases disabling the safemode for fuse bits.
- *-t* – uses interactive terminal mode instead of up or downloading files.
- *-v* – Verbose
- *-U memtype:op:filename[:format]*
  - *memtype* – Memory types are
    (i) *calibration* – One or more bytes of RC oscillator calibration data.
    (ii) *eeprom* – The EEPROM.
    (iii) *efuse* – The extended fuse byte
    (iv) *flash* – The flash ROM of device
    (v) *fuse* – The fuse byte in devices with a single fuse byte.
    (vi) *hfuse* – The high fuse byte.
    (vii) *lfuse* – The low fuse byte.
    (viii) *lock* - The lock byte.
    (ix) *signature* - The three device signature byte (device ID).
  - *op* – Operations are
    (i) *r* – Read the specifed device memory and write to specified file.
    (ii) *w* – read the specifed file and write to specifed device memory.
    (iii) *v* – read the specified device memory and the specified file and perform a verify operation .
  - The *filename* can be either a fileName, immediate byte value (in decimal, binary,hexadecimal, etc)
  - *format* is optional and can
    1. *i* - Intel hex
    2. *r* - raw binary
    3. *e* - the elf files
    4. *m* - immediate mode
    5. *d* - decimal
    6. *b* - binary(0b)
    7. *d* - hexadecimal(0x)

8.3 Example

8.3.1 Downloading hex file into device Flash

```
avrdude -p atmega8 -b 19200 -c stk500 -p /dev/ttyUSB0 -v -U flash:w:hello.hex:i
```

8.3.2 Uploading Flash from device into file

```
avrdude -p atmega8 -b 19200 -c stk500 -p /dev/ttyUSB0 -v -U flash:r:"./readFashMemory.bin":r
```

8.3.3 Reading Device signature

```
avrdude -p atmega8 -b 19200 -c stk500 -p /dev/ttyUSB0 -v -U signature:r:"deviceSinature.text":h
```

8.4 Writing the High fuse

```
avrdude -p atmega8 -b 19200 -c stk500 -p /dev/ttyUSB0 -v -U hfuse:w:0x65:m
```