AVR

- Revision

Revision

- History (Alf Vegard)
- AVR core
- ATMEGA32
- Registers

AVR is RISC

- AVR is Harvard architecture.
    - Buses and address spaces of data and instructions are different.
    - Modified architecture i.e. Read only data can be stored in Flash.
- Many GPR -- AVR have 32 GPR.
- Instruction size is same.
    - AVR most of instructions are 2-bytes long.
    - Few instructions are 4-bytes.
    - AVR flash memory each location is 2-bytes (it can contain 1 instruction).
- Pipeline
    - AVR has two stage Pipeline.
    - Fetch and Execute.
- Small instruction set -- micro-instructions.
- Most of instructions execute in single cycle.
- Load-store architecture
    - LD & ST instructions.
    - Pre and Post increment modes are supported.

AVR programming

- Can be programmed in Assembly language.
- Can be programmed in C language.
- AVR compilation is always cross-compilation.
- Toolchains
    - ATMEL Studio
        - IDE, Compiler, (ATMEL) Assembler, Linker & other tools.
        - Includes simulator for testing assembly code.
        - It is available only on Windows.

    - AVR GNU Toolchain*
        - Compiler, (GNU) Assembler, Linker & other tools.
        - It is available for all platforms.
    - VMLab IDE*
        - IDE, Compiler, (ATMEL) Assembler, Linker & other tools.
        - Includes simulator for testing assembly code.
        - It is available only on Windows.
        - Can be installed on Linux (on WINE).
- AVR programmers
    - To burn program (hex file) into flash.
    - FlashMagic -- on windows
    - avrdude -- on all platforms.

AVR address space

- AVR have different address space for data and code.
- AVR code address space -- mapped to flash ROM.
    - user program space
        - Begin from 0000h
        - Depends on size of flash. E.g. ATMEGA32 have 32KB flash.
    - bootloader space (optional)
        - Available in ATMEGA128.
- AVR data address space
    - AVR registers
        - R0 to R31
        - address range: 0 to 31
    - IO address space
        - 64 bytes IO addresses (for ATMEGA32)
        - address range: 32 to 95
        - 160 bytes of additional IO address space is available in advanced AVRs (e.g. ATMEGA128).
    - RAM
        - 2KB of RAM in ATMEGA32
        - address range: 96 to 2143
        - External RAM support is available for few AVRs.

AVR Assembly Programming

- AVR stack
    - AVR SP is 16-bit i.e. SPH + SPL.
    - SP is in IO memory (IO addr: 0x3D & 0x3E) (Mem addr: 0x5D & 0x5E).
    - AVR follows Empty Descending stack.
    - Its SP should be initialized to the end of RAM.
        - end of RAM = registers (32) + IO registers (64) + RAM size (2048) - 1 (ATMEGA32)
    - AVR Assembler provides a special macro/variable RAMEND that is auto set to RAM end address depending on AVR controller.
- AVR stack instructions
    - PUSH Rs
        - Push value of Rs on stack and decrement SP.
    - POP Rd
        - Increment SP and Pop value into Rd.
- HIGH() & LOW() macro
    - AVR Assembler provides built-in macros to separate higher and lower byte of any 16-bit number.
- SREG
    - N - Negative flag
        - When result is negative (for signed number calculation)
            - N = bit7 (if result is signed).
    - Z - Zero flag
        - When result is zero
    - C - Carry flag
        - When carry is generated from bit7 (usually for unsigned calculation)
    - V - Overflow flag
        - When carry is generated from bit6 (for signed calculation) but no carry from bit7.
        - When carry is generated from bit7 (for signed calculation) but no carry from bit6.
    - H - Half Carry flag
        - Carry from lower nibble (bit3) to higher nibble (bit4)
    - S - Sign flag
        - N ^ V = S

        - 0 ^ 0 = 0
        - 0 ^ 1 = 1
        - 1 ^ 0 = 1
        - 1 ^ 1 = 0
    - I - Interrupt Enable/Disable
        - I=1 -- enable interrupt -- SEI instruction
        - I=0 -- disable interrupt -- CLI instruction
    - T - Bit copy storage
- Jump instructions
    - Change PC to the new address.
    - JMP label
        - 4-byte instruction
        - label address is 0 to 4M
        - 3 cycles
    - RJMP
        - 2-byte instruction
        - Relative Jump -- w.r.t. current PC.
        - address (offset) can be +/- 2K.
        - 2 cycles
    - IJMP
        - 2-byte instruction
        - Indirect Jump -- address from Z register is copied in PC.
        - The address range is 0 to 64K.
        - 2 cycles
    - BRxx
        - 2-byte instruction
        - Conditional Jump -- jump only if condition is true.
        - 2 cycles
            - 1 cycle to check the condition
            - 1 cycle to jump
        - Relative Jump -- w.r.t. current PC.
        - address (offset) can be +/- 64.
- Function instructions
    - All CALL instructions push address of next instruction (return address) on the stack and jump to the function address (PC).
    - CALL
        - 4 bytes instruction
        - address range: 0 to 4M
        - cycles: 4 cycles

  - RCALL
    - 2 bytes instruction
    - address range: +/- 2K
    - cycles: 3 cycles
  - ICALL
    - 2 bytes instruction
    - address range: 0 to 64 K -- address in Z register
    - cycles: 3 cycles
  - RET
    - To return from function
    - Pop return address from stack and load in PC
    - 2 bytes instruction
    - cycles: 4 cycles

ATMEGA32 GPIO

- ATMEGA32 have 4 ports each of 8 pins -- PA, PB, PC, PD.
- For each port there are 3 registers
  - DDRx -- Data Direction Register
    - DDRx.n = 0 --> Input pin
    - DDRx.n = 1 --> Output pin
  - PORTx -- To write on port pin
    - PORTx.n = 1 --> Pin voltage HIGH
    - PORTx.n = 0 --> Pin voltage LOW
  - PINx -- To read from port pin
    - Pin voltage HIGH --> PINx.n = 1
    - Pin voltage LOW --> PINx.n = 0

To program ATMEGA32

- AVR EmbeddedMarket board
  - terminal> avrdude -c usbtiny -p atmega32 -U flash:w:main.hex
- AVR RhrydoLabz board
  - terminal> avrdude -c avr109 -p m32 -P /dev/ttyUSB0 -v -V -U flash:w:main.hex