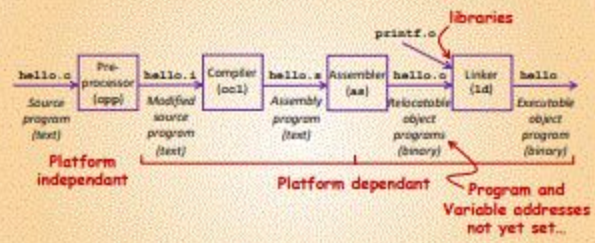
Compiling and Running a C Program:

```
Unix> gcc -Wall -o hello hello.c
```



➢ **Execute your program** by typing the name of the executable at the Unix Prompt:

```
Unix> hello
```

Compile time - Vocabulary:

- **Preprocessor**: prepares file for compiler, handles duties like processing macros, sources selection, processing preprocessor directives (indicated by # in C) and file includes
- **Compiler**: converts nearly machine independent C code to machine-dependent assembly code
- **Assembler**: converts assembly language to machine language, but result is a re-locatable object file, meaning addresses of code and variables have not all be resolved
- **Linker**: combines all object files and resolves addressing issues among them and determines final addresses for code and variables
- **Loader**: when we execute the program, loads the executable file into memory - it makes sure that main function is in a address that reflects the start of program.
  - More details on linker-loader:
    - http://www.harbeck.org/linkers/linkers.html
    - http://www.linuxjournal.com/article/6463
- **Cross Compiler**: compiler that runs on one platform but outputs code for another target machine (our AVR code is compiled Intel Processor)

AVR IO Ports – Programming I/O Ports –Assembly - pg1

```
;Using CBI and SBI to write to ports
SBI DDRB,  3 ;make bit 3 as output bit on PORTB
CBI PORTB, 7 ;make PORTB bit 7 as "0"
SBI PORTB, 4 ;make PORTB bit 4 as "1"

;Using OUT instruction to write to ports
LDI R18, 0b00100000
OUT DDRB, R18     ;make bit 5 as output bit on PORT
LDI R18, 0b00000000
OUT PORTB, R18    ;make PORTB bit 5 as "0"
LDI R18, 0b00100000
OUT PORTB, R18    ;make PORTB bit 5 as "1"

;INPUT EXAMPLE
IN R18,PINB
```

A common error here is that OUT DDRB, R18 doesn't just set bit 1 to a "1", it also sets all of the other bits to "0", so please use the following method to set bits instead

Types: - Integral Data Types:

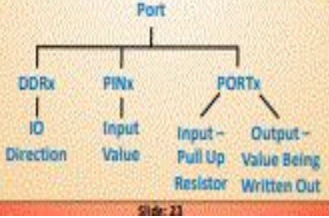
- C data types for storing integer values are
  - **int** (the basic integer data type - int should be used unless there's a very good reason to use one of the others)
  - **short int** (typically abbreviated just as short)
  - **long int** (typically abbreviated just as long)
  - **long long int** (C99)
  - **char** (C does not have "byte")
- Number of Bytes
  - **char** is stored in 1 byte
  - The number of bytes used by the other types depends on the machine being used.

Types: - Integral Specifiers:

- Each of the integral types may be specified as either
  - **signed** (positive, negative, or zero)
  - **unsigned** (positive or zero only) (allows larger numbers)
  - **signed** is the default qualifier
- Much more on this later…
- Note: be sure to **pay attention to signed vs. unsigned** representations when transferring data between system. **Don't assume.**

AVR IO Ports – Notes (cont - see diag Slide #4):

- If **DDRxn** is written logic one, Portxn is configured as an output pin. If **DDRxn** is written logic zero, Portxn is configured as an input pin.
- If **PORTxn** is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).
- If **PORTxn** is written logic one when the pin is configured as an input pin, the pull-up resistor is activated.
- **Special Feature note**: writing a logic one to a bit in the **PINx** Register, will result in a toggle (inversion) in the corresponding bit in the Data



Slide: 23

AVR IO Ports – Programming I/O Ports –Assembly – pg2

```
;set pin 4 of B port as output
; without affecting other bits
IN R18,DDRB
ORI R18, 0b00010000
OUT DDRB, R18

;set pin 5 of B port to 1
; without affecting other bits
IN R18,PORTB
ORI R18, 0b00100000
OUT PORTB, R18

;clear pin 4 of B port to 0
; without affecting other bits
IN R18,PORTB
ANDI R18, 0b11101111
OUT PORTB, R18
```

Toggling

```
;toggle pin 1 of B (no eori available)
; without affecting other bits
IN R18,PORTB
LDI R19,0b00000010
EOR R18, R19
OUT PORTB, R18

;toggle pin 1 of B using PINB "input
; write trick"
OUT PINB, 0b00000010
```

Changing Multiple Bits

So, sbi and cbi are more convenient and allowed if only changing one bit at a time. If multiple bits need to be set at the same you can't use sbi,cbi

```
;set pin 7,3 of B port to 1 at same time
; without affecting other bits
IN R18,PORTB
ORI R18, 0b10001000
OUT DDRB, R18
```