In the beginning

Fetch Execute Cycle (FEC)

ALU: "Registers"

Bob, with purpose & name

# of general purpose registers is $2^n$ because they must be referenced in instructions with N bits. On MSP 16 registers

This simple picture omits:

1) I/O
2) Interrupts
3) Transfer of control to different stored program (and back)

We have a specific instance of above with MSP

CPU - 16 registers

RO

R1 {Hardware dictates}

R2 {purpose}

frame pointer

function

arguments

return

FEC

CPU

STACK POINTER

STATUS/CG

CG

SEQUENTIAL

ADDRESSED

BOBs

PDIR, PIOUT, etc.

Free on MSP with

Specific Registers

RAM, I/O, CODE

Program counter

Software tool

Chain dictates

R12

R15
When we create a stored program we begin with source code(s) which are then translated into object files which are then compiled/ assembled linked/loaded to create an executable which can be placed in memory as a running program on (Unix) process

Unix (file descriptors)
Every process has 3 I/O associations
- Standard input, output, error which are initially the 'console' (screen, keyboard)
These can be redirected or piped >, <, >>

Every process has a
- Current working directory
- Search path (list of directories)
- Process id

Bash is a shell or command line interpreter which executes commands which are programs on specific path e.g. $/bin ls or on searchpath $ls where /bin is in list

. /g.out is needed because . is not on search path
Bash commands: ls, cat, cp, mv, alias, cd, man ...(many)

The file system is hierarchical with / the root directory
and file referenced PATH / FILE
Path can be absolute /bin/ls or relative .. /bin/FILE
with . , .. current or parent

When a user logs in their current working directory is their
home directory

C programs are translated with C++ then compile.

cpp - #include, #define + others

Then a two pass compiler we use gcc, msp430gcc

gcc - c creates object file only

make allows automation (makefiles)

Structure of C program

includes
prototypes of your functions
int main (int argc, char* argv[])
{
    Q = func (2V)

    }  \quad \text{pass by value}

    \text{Get reference via \texttt{ptr}}

    \texttt{<type> Func (int * V}

    \}  \quad \text{pass by value}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}

    \}