Toolchain

- **CPU** is brain of computer.
- Binary language understands a computer.
- Microcontroller is powerful bcoz they work a specific.
- Processor is generic (CPU menses processer. CPU is the important).

Why CPU understands binary language.

ANS: कारण CPU हा electricity वर चालतो मानून त्याला दोन च अवस्था असतात एक तर चालू किंवा बंद मानून binary language understand .चालू म्हणजेच one आणि बंद म्हणजेच zero.

Binary Digit

Bit (आत सगळे खेल तर number वरच असतो).

64 architecture म्हणजेच 64 wired

1.Instruction set : Processor म्हणजे hardware त्याला binary च समजते .(binary programming) processor त्याचा instruction set देतो त्याच नुसार त्याला instruction आल तरच तो work करतो . (processor wise त्यांचे instruction set हे वेळगे वेगळे असतात .)

2.Converter :(convter Software आहे हा का तेर decimal numbers कळतात) हा Decimal number ला binary मध्ये convert करतो ।तो त्याला OPCODE मानतात । OPCODE mense operation करणारा code। (user कडून decimal number घेऊन त्याचा instruction set मध्ये बगुन तो त्याला तो binary मध्ये convert करुन processor कडे पाठवतो

3.Assembler : आत्ता आपल्याला word मध्ये बोलाच like .Subtraction , Addition असं बोलायचं त्याचं language ला assembly language मानतात . Assembler हा translate करतो word to opcode to binary. Assembly language is not portable its processor specific. Processor नुसार ती change होते. त्याच्या instruction set ला Nemonic मानतात. Intel , RMD या दोन processer company आहेत याचा दोनच पण instruction set वेगळाच असतो तो पण version wise सुद्या वेगळ असतो . 64,32 bit wise . .

Storage : processor ला specific च जागा दिसते म्हणजे (Intel X-86 processor. 440 bit) ची जागा दिसते . त्याची नजर येवडिच आहे याच्या पुढे काही दिसत नाही .आज आपण जे operating syatem ठेवतो 440 bit च्याच जागी ठेवतो .त्यालाच bootable device पण मानतात जे आपली os रोज चालू होतोना boot होते .OS install करताना म्हणजेच windows or linux ची

file system install करतो असं आहे ते . BIOS Basic input output software. <- इथे Bootable ची वगरे setting असते .

पहिलचे language हे सगले types less language होते तिथे data types वगरे नवथे .त्यानी B language ही brian karnighan यानचीं होतीं.

हे सगळे problems येत आहेत महँनून Dennis Ritchie C language बनवाली त्या मध्ये data types वगरे add केले त्या आधी कोणत्या पण language मध्ये data types नवथे मग C Language use करुण त्यानी **unix operating system rewrite** केली १९७३-७४. तेहाँ **c** language आली तिला high level language पण मानतात.

Features of c language:-

1. General purpose language also system programming language

2. Code portable.

3. Processor oriented(C , CPP)

1. LOW LEVEL – BINARY LANGUAGE

2. MIDDLE LEVEL – ASSEMBLY LANGUAGE

3. HIGH LEVEL – C , C++, JAVA …

COMPILER :

आता आपन थोडस **compiler** विशइ जानून घेउ . COMPILER हा high level to middle level convert करतों .

COMPILER 5 PHASES. आहेत.

1. Lexical Analyser :- ha अपला c चा code वेगवेळ्या Tokenize करतों.(अपल्या code च्या partition-wise मध्ये त्याचे टुकड़े करतों)

2. Syntax Analyser :- अपला code हा योग syntax चा आहें का ते check करतों .

3. Semantic Analyser :- ईथे आपण जे कही लिहल आहे ते सगल right आहें कि नाही है check होते .

Like ex : 10 % 0 ही error आहें असं ईथ check होत.

4 :- Code Optimizer : ईथे आपला .c code हा कामी किंवा जास्त करण्याचं काम होते.

like : int ans= iNo1 + iNo2;

return ans;

Optimize नंतर :

return iNo1 + iNo2;

आस हे code optimize होते.

आजून तसच Time & Space Complexity कमी करता येते का यची पण कलाजी घेतली जाते .

Time Complexity No of iteration कमी करणे.

Space Complexity No of Variable कमी करणे.

5 . code generation :- या phase madhe processor la कलनारा assembly चा code geneate होतों हा code त्याच dekstop पूर्तताच असतो का तर assembly is not protable . Specific that machine.

Compiler कायकाय लागतें .

Compiler ला only declaration लागतों .defination चीं गरज नाही .पण दिले तरी नको मानत नाहीं . तू जे जे तुज्या code मध्ये वापरनार आहेस त्याच त्याला declaration लागतें .

आपण आता proper toolchine चे tools बगनार अहोत.

A) Editor : त्यात आपण c programmer लिहणार .

B) pre-processor :- compiler आधी तुजा program हा pre-processor कडे जतो .तिथे तुजा program माथले # वाले सगले काम होतत .

Header files मथे only declaration च असतात. Stdio.h ही library नाहीं ही header file ahe यात सगले input output statement चे declaration आसतात.

याचे definition हे libc या library मध्ये आसतात. या library सगले code हे binary from मध्येच असतात . का तर ते slandered आहें कोनी पण येऊन change करू नये म्हणून.

आता आपन या tool कोण बोलवतो है बगु तर . ते आहें gcc or cc हे compiler नाहित .compilation चे tool आहेत हे gcc or cc हे एका माघे माघे सगळेच tools बोलवतो .

Pre-processor हा वरुण ख़ाली program ला parsing करत करत जातो .यालाच top to bottom parsing आसें पण म्हणतात .

च्या पुठच्या world ला मला Direct कर आस मनतो त्यालच pre-processor directives म्हणतात . .

Pre-processor चे काम

1) #include < > or " "

< > Angular bracket मध्ये जे headerfile आसते ति preprocessor माहिति असण्याऱ्या include directory मध्येच शोधतो. Computer double click -> properties-> advance system setting-> environment variable -> include directories मध्ये बगतो . Pre-processor .c file चि copy बनवातो आनी त्याला include directory मध्ये ति header file थेटल्यावर त्यातला totol code copy करुण सोत्याच्या file त्या line paste करत जातो. Operating system त्या setting सगली processor देतो.main ला तीन parameter आसतात .(argc ,argv, environment variable)

""double quote मध्ये जे header file आसते ति अपल्या cerrent directories मथें आसते ते आपण सोता write केलेली आसते .h extension देउन बनवाली जातें.

Header file मध्ये only declaration असते कारण ते Definition हे c Source file मध्ये ढेवावच लागतो client देताना आपण c file नाही देतो object file देतो . त्या मुळे Header file मध्ये declaration असतात का तर function parameter कळावं लागते .त्या मुळे

2) #define micro expansion

3)# pragma हे एकदी size फ़िक्स धेवनया करीता पण होते like structure मथें use करता येते.

4) #if #endif आशे conditional compilation साठि पण use होतो. म्हणजेच conditional deside होतो. सागला code लिहलेल आसतो पण तोच specific code Run होतों condition नूसार .

5) Unwanted space पण कड़तो .

Pre-processor हा output file म्हणून .i file देते त्यालाच intermided file पण म्हणतात.

C) Compiler: कड़े सोतच Data structure table आसतो त्याला symbol table म्हणतात. तो जे वाचत जतो ते symbol table मथें entry करत जातो. मग ते सगले declaration right अस्तिल तर त्याचि तो assembly code generate करतों ते पण definition अस्तिल तरच तो assembly मथें अस्तितव देतो .asm तो symbol table destroy पण करतों .

All global variable, All local variable, All function मथल code यानच memory असते.

NON Executable : all declarations, all variable definitions also Non executable statement.

Executable statement: Function मधलाच काम कारनार code तोच executable code.

Memory च संबध नाही Executable Non Executable statement शी.

Rule: Non Executable code must be appear before executable statement inside block .

3) Assembly:

तीन च गोस्टी आसतात local variable(तो code बनवलेला), global variable, function code (Executable code) .

. c File covert compiler to assembly	Section
1. GLOBAL VARIABLE	1 . initialize (.DATA SECTION (NON-BSS)
	2. Non- initialize (BSS)
2 FUNCTION CODE (executable code)	3.TEXT (READONLY)
3.STRING LITTERAL CONSTANT	4 .RODATA
4 .LOCAL VARIABLE (proper not available read msg…)	.COMPILE GIVE BY TEXT SECTION BY SELF COMPILER CREATED.

3.STRING LITTERAL CONSTANT : " " मध्याला STRING LITTERAL CONSTANT ला ASSEMBLY मध्ये मेमरी मिळते.का तर STRING LITTERAL CONSTANT घेणारा POINTER असेल तर EX : char *p ="hello" आशा गोष्टीच ना ASSEMBLY मध्ये मेमरी मिळते ते पण RODATA मध्ये का तर ते change होत नाहीत का तर ते scanf printf मध्ये use करतो .आणि यांचं पहिला parameter हा constant असतो म्हणून.

4 .LOCAL VARIABLE : कुठे जातात तर .local variable हे limited of that function पुरतेच असतात .मग compiler अशा local variable साठी सोता code generate करतो केंव्हा जेंव्हा call होईल तेंव्हा मेमरी देता येईल .program Run त्याच threshold point त्याचा खाली त्या variable च्या size ने खाली ठेवतो तस तस पुढचे पण variable तसेच ठेवत जातो जस call तस तस त्यांना मेमरी मिळत जाते हा code TEXT section मध्ये ठेवतो call जाल कि येतो मग .असं compiler local variable manage करतो .

ASSEBLER काय काम करतो :

ज्यांना ज्यांना नाव आहेत त्यांना तो ADDRESS देतो ते पण randomly देतो .त्या साठी त्याच्या कडे त्याचा symbol table आहे त्यात ते addresses store करतो.Address का देतो तर linker ला address लागतो मानून देतो म्हणून म्हणतात कि assembler given by address that address is static मेमरी allocation.

.asm or .a त्याची binary from मध्ये convert करतो मंजेच object from ex : .o मध्ये.

4.LINKER : linker is operating system specific.

linker ला address लागतात . ते address हे assembler देतो (त्या symbol table मध्ये टाकून) यालाच virtual address पण म्हणतात .

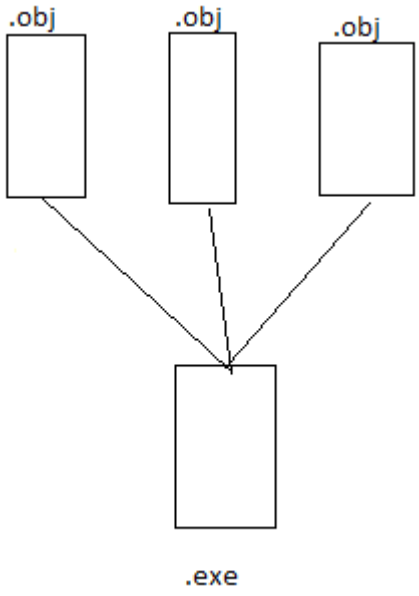
symbol table जर address नसेल तर तर error देतो.

linker object file link करतो मनजेचं तो सगळ्या object file एकत्र करतो. ex printf , scanf यांचं code आपण लिहत नाही म्हणजे compiler कडे फक्त (symbol table) मध्ये त्यांची ओळख आहे . मग printf , scanf linker libc.lib हे file आपल्या object सोबत लिंक करतो म्हणजेच त्या file च्या symboll table मध्ये त्यांच्या Entry आपल्या file मध्ये करतो .

त्या file चा text आपल्या file मध्ये टाकतो

data पण merge करतो .

जेव्हा दोनी symboll table एकत्र येतात तेंव्हा address duplicate होऊ शकतात मग जर असं झालं तर linker ते check करून सगळे address हे UNIQUE मिळावे याची काळजी घेतो .



MOST IMPORTANT

जर तुज सगळं बरोबर असेल तर त्या मध्ये तो अजून एक गोस्ट add करतो ती म्हणजे primary header त्या मध्ये सगळी information असते .

1. Primary header (magic number)

2. Address of entry point function म्हणजेच तुझा object file मधून text मधून कुठल्या function पासून execution start करायच हे सांगतो म्हणजेच Address of entry point function म्हणजेच Main हा entry point function नाही..... Main नाही तर मग कोण आहे _START (linux) (main CRT WINDOW) हा आहे का तर जेंव्हा आपले object file इकत्र येत असताना तेंव्हा CRT (C RUNTIME LABARARY) या नावाची LABARARY मिळते ते पण object fome मध्ये या labaray मधून _start वरून main ला call येतो main हा entry point function आहे पण ते user define function मधला entry point function आहे.

3. Time date stamp.

4. Types of executable: 01.self-executable (Double click करून open होणारी)

02. Depended executable (जो परेत कोणी बोलवत नाही तो परेत Run होत नाही)

जेव्हा आपण executable run करतो तेव्हा operating system चा loader येतो आणि त्या executable primary header बघून information check करतो .

जर सगळे tools बगायचे असतील तर त्याचे command आहेत (linux os 32 bit)

cpp myfile.c -o myfile.i //cpp म्हणजे c pre-processor

cc -S myfile.i -o myfile.s // compiler

as myfile.s -o myfile.o //assembler

ld myfile.o -lc -dynamic-linker/lib/ld-linux.so.2 -o myfile -e main // loader links

./myfile //output

5. LOADER :

LOADER हा कधी picture मध्ये येतो तर जेंव्हा exe वर double click करून तेंव्हा process होऊन RAM वर आण्याचं काम करतो त्याला Loader म्हणतात .

तो आत्ता local variable आणि function parameter जो code .text Section मध्ये होतो ते आत्ता stack frame मध्ये येते. तो त्याचं function पुरताच असतो . अजून तसेच बरेच येतात .DATA(bss, non-bss) .TEXT ,HEAP(dynamic memory), RODATA ,STACK FRAME , हे सगळं आता वर होणार RAM . थोडंस stack and heap च बोलू . म्हणजे काय असते तिथे .

Stack (stack frame **):** जेव्हा process Run होते तेंव्हा stack frame बनते प्रत्येक process सोताची stack frame असतो .

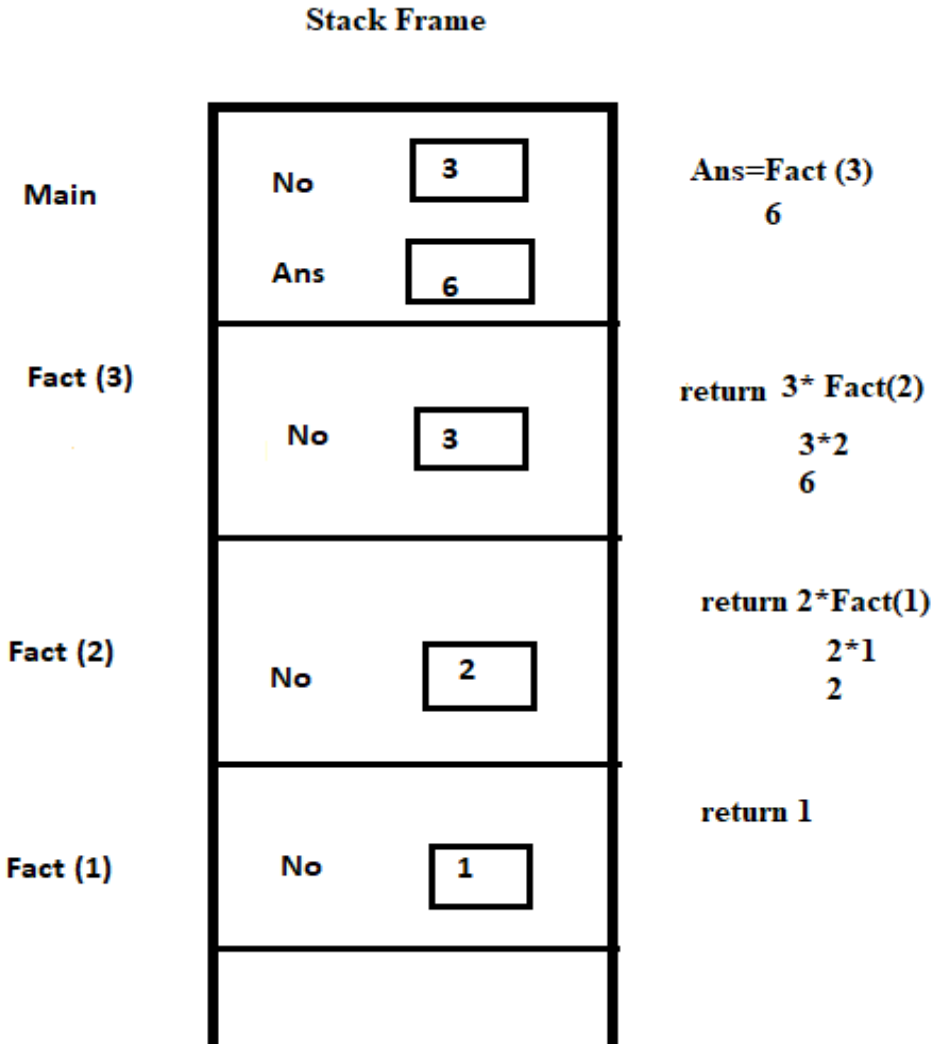
त्या stack मध्ये stack frame जातात stack frame प्रत्येक function ची असते function call झाला कि frame push होते ,function संपला कि ती frame pop होते .stack हा top to down grow होतो.

stack frame मध्ये काय असतो ?

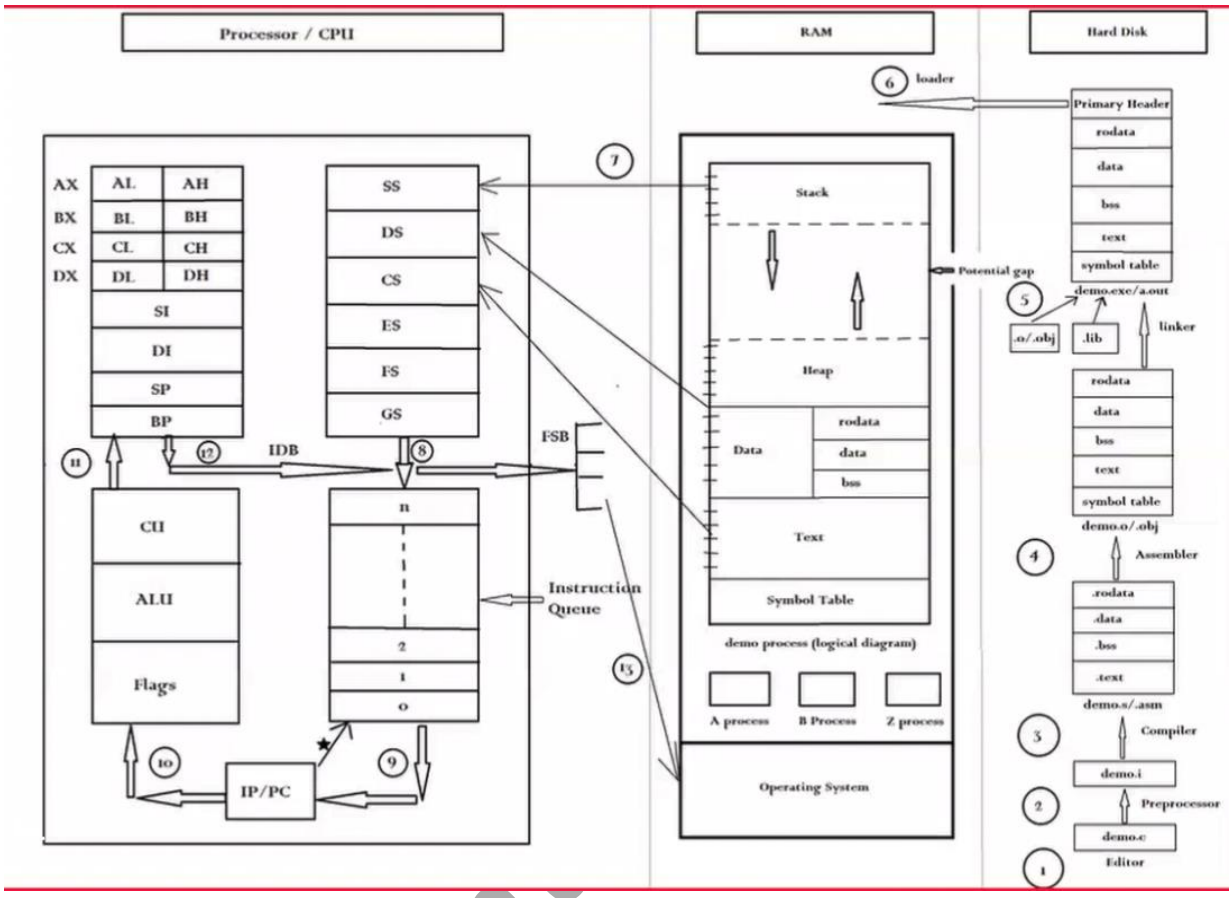
stack frame मध्ये सगळे local variable असतात .

तिथल्या memory मध्ये value ही garbage असते का तर तिथली memory ही zero-out होत नसते .

Heap: heap momory runtime ला assign होते dynamic momory allocation झाल्यावर momory मिळते.(malloc, calloc , realloc).



One program flow one by one :



Processor side:

Multitasking म्हणजे काय ?

एकाच वेळी अनेक काम होत आहेत याच FEEL देणे .

Multitasking programing करता येत नाही . कुठ्यालाच programing syntax नाही तास just theoretical word आहे.

Multitasking types :

1.multitherading : एक काम अनेक threads मिळून करणे . ex: एकटा कार चालवणे.

2.multiprocessing : एक काम अनेक process मिळून करणे. Ex: drawing class तुझा हाता मध्ये streing and शिकवणारे च्या कडे break and gheres .

Register : 8/16/32/64 bit wise च असतात जेवढे bit तेवढे wire यांचंच collection ला BUS म्हणतात .

1. Data bus : Read write काम होतात

2. Address bus. कोणत्या address काम कराच ते सांगतो.

3. Control bus हा सांगतो Read कराच का write .

Types of Register :

1. Segment Register

1. Core Segment
2. Data Segment
3. Stack Segment
4. Extra Segment
5. FS
6. GS

या registers मधुन आपला Code transfer होत असतो. हे registers (FS , GS) कमी पडलं तर Last चे use दोन करतातं.

2. General purpose register:

1. AX (Accumulator) (16 bit) , EAX(32) ,RAX(64) :Input output Related काम करतात , Arithmetic काम करणे , EAX काही तरी return करतो function ला ती value या register मध्ये store होऊन जाते

2. DX (Data) : AX जर कमी पडला तर त्याचा सोबत DX use करतात. Ax ला help करतो याना(AX , DX) जोडीने use केला जातो.

3.CX (Counter) : looping चे काम असतील तेंव्हाच याचा use होतो . Counter ची value store करण्या साठी.

4.BX (Base): हा जेंव्हा Address असतील तेंव्हाच use होतो

3. Index register:

1. Source index (SI) : string programming both are use ex: copy string source and destiny like .

2. Destination index (DI)

4. Flag register : हा तसा एकटाच आहे पण तो आपल्या Architecture एवढे flag set करू शकतो (ex : 64 bit ,64 Flag set)त्यांचे नाव आहेत कशा साठी कोणता use करावा या साठी

5. pointer register : या Register मध्ये Addresses store होतात .

1. Instruction pointer: Next instruction of address.(virtual address to physical or fetch to execute cycle)

2. Base pointer: sp पुढचं address(instructions) घेतो तेंव्हा sp मधला address हा Bp store करतो का तर नंतर त्या function च काम झालं कि ती stack frame जाते पण तेंव्हा तो कुठे point कराच हा problem झालं असत म्हणून तो old Address हा BP store करून ठेवतो.

3. Stack pointer