Mybcafill.sys

Primary	Derived	User-define
Integer	Array	structure
character	Pointer	Union
single presion floating point	Function	Enumeration
Double presion floating point	~~Enumeration~~	~~Function~~
§ Void		

operator compile time work hotat.

Vidyalekhan
DATE
PAGE

identifire rules

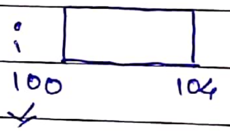
int i; // allowed
double d; // allowed
void v; // not allowed
why?
Bcz void doesn't have any size.

sizeof(void); // 1 byte.
- 'sizeof' is operator not function.
kiti mlm milali he nai sangat
kiti mlm milel he sangto.

size & memory (differentiale)

int i;
mlm ahe — i la size ahe, mlm pn ahe

mlm mhanje bhand.
size mhanje bhand kewdha



mlm — 4byte : size

jycha jycha pointer banwata yeto te sagla datatype asta. ani void cha pointer banwata yeto (function cha pn pointer banwata yeto)
void *p;

Preprocessor
input: abc.c
① tyachi ek file banwato tyat .c → contents takto
~~#---~~ #include< > , #include" "
File shodh, sapadli → content copy kar → tyachya file madhe peast kar

Preprocessor directive

define
include
pragma
IF
IFDEF

1 Pre-Processor,

pre-processor editor kadun '.c' File gheto, ani tychi swatahachi ek File banwato ani tyat '.c' le sagle content as it is copy karto, ani ata ya File war tyche kam krto.

File inclusion — #include <stdio.h>
Macro Expansion — #define MAX 10
Conditional compilation — ~~#IF~~, #ifdef, #else, #endif
Remove unwanted space & line

```
#define MAX 10
#ifdef MAX
        // code
#else
        // code
#endif
```

Ya case mdhe 'ifdef' चा code work hoil ani else चा code remove hoto ani 'ifdef' चा code samor compiler kade jato.

- # include <stdio.h>

'stdio.h' he header File shodh, tytle content copy karto ~~tuzya~~ tychy File mdhe sagle.

- #define MAX 10

code mdhe jithe jithe MAX ahe tithe tithe 10 replace kara.

2 Compiler

① Lexical analysis
② syntax analysis
③ semantic analysis
④ code optimization
⑤ code generation

- Compiler la saglyanchi olakh lagte, ie declaration. Jya gashtinchi tyala olakh naste tyachi to error dete Undeclare ---;
- Compiler line by line vachat jato top-to-down approach.

To je je vachto tychi entry symbol table mdhe karto.

Symbol Table

Name	size	value	address	To	From

Name : navat a tyacha, nav lasta.
int, i, cont, etc...

Size : integer asel tar 4 byte

value : variable identifieres nach fkt value aste, date je assign kela te navtar tyat nai.

address : jar fkt declaration asel tar, 'address' milat nai. definition asel tarch address milte address linker la lagte, so address the compile or linker he doghach detat.

To, From } Tya identifier cha scope kuthun kuthaparyant.

- Declaration or Definition jar asel tarch symbol table la hat lawla jato.

```
extern int i;  // declaration
int i;  // definition
```

Compiler is processor specific
Linker is operating system specific

Vidyalekhan
DATE
PAGE

int i = 10;
Definition ahe, & symtab madhe 10 he value attach tak lyan jail.

int i; // ithe 'i' chi entry hoil, address milel value nahi.
// code
i = 10; // ya line la symtab la hat lawla jaanar nahi, assigment ahe. executable statement.

Executable statement, runtime la execute hotat.

Non-executable statement, compile time la baghitle jatat.

Rule:
Non-executable statement must appear before executable statement inside block.

Compiler saglyachi olakh symtab mdhe theuto ani # '.asm / .s' file generate kato, Ya file mdhe assembly code asto, ani file assembler la deto.

- Loader is operating system specific.

Vidyalekhan™
DATE
PAGE

3 Assembler :

- Assembler aleli .asm l.o file ghete, tytla assembly cha code 'binary' code mdhe convert krto ani .obj l.o file generate krto. Ani linker la pass krto.

Binary code == object code == machine understandable code.

4 Linker

Linker la address lagto.

- Linker symtab la baghto ani jar addr nasel tar error deto.

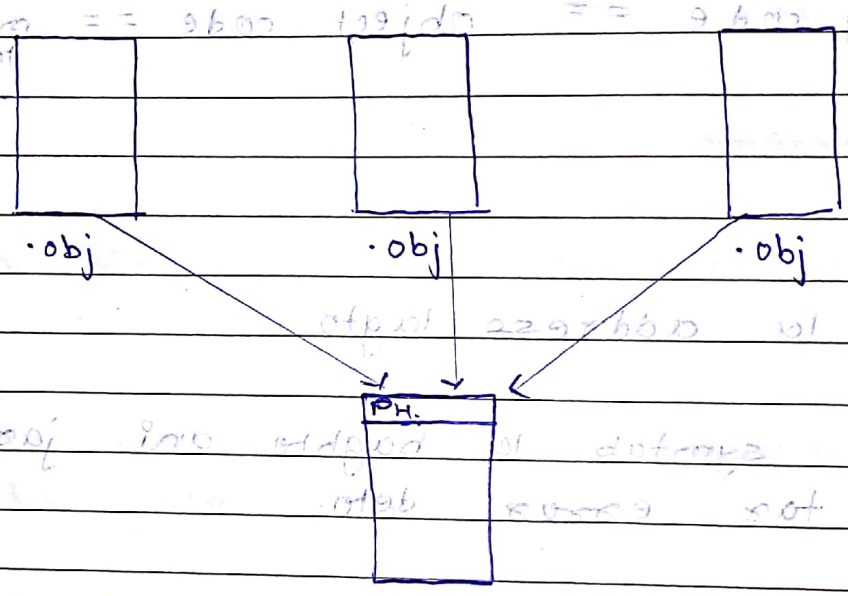
- Linker object file link krto, mhanje to 'saglya' object file ektra krto.

ex: printf, scanf chya code apan lihat nai, pn tya mhanje compiler kde fkt (symtab) mdhe tyachi olakh make.

mg linker libc.lib he file aplya .obj sobat link krto mhanjech,
- tya file cha symtab to mdhlya entry aplya file mdhe takto
- tya file cha text aplya file mdhe takto
- Data pn merge krto.

Jeva don symtab extra yetat, address duplicate hou shakto---!

Mag, jar nas zala tar linker mate check karun saglyanna UNIQUE address milel yachi khatri ghete.



Most Important.

Linker ne executable file war Primary Header add karto.

Primary Header mdhe 4 goshti astat:

① Magic no
② Address of entry point
③ Time - Date stamp
④ Type of executable → i) self executable
ii) dependant executable

Executable run keli,...

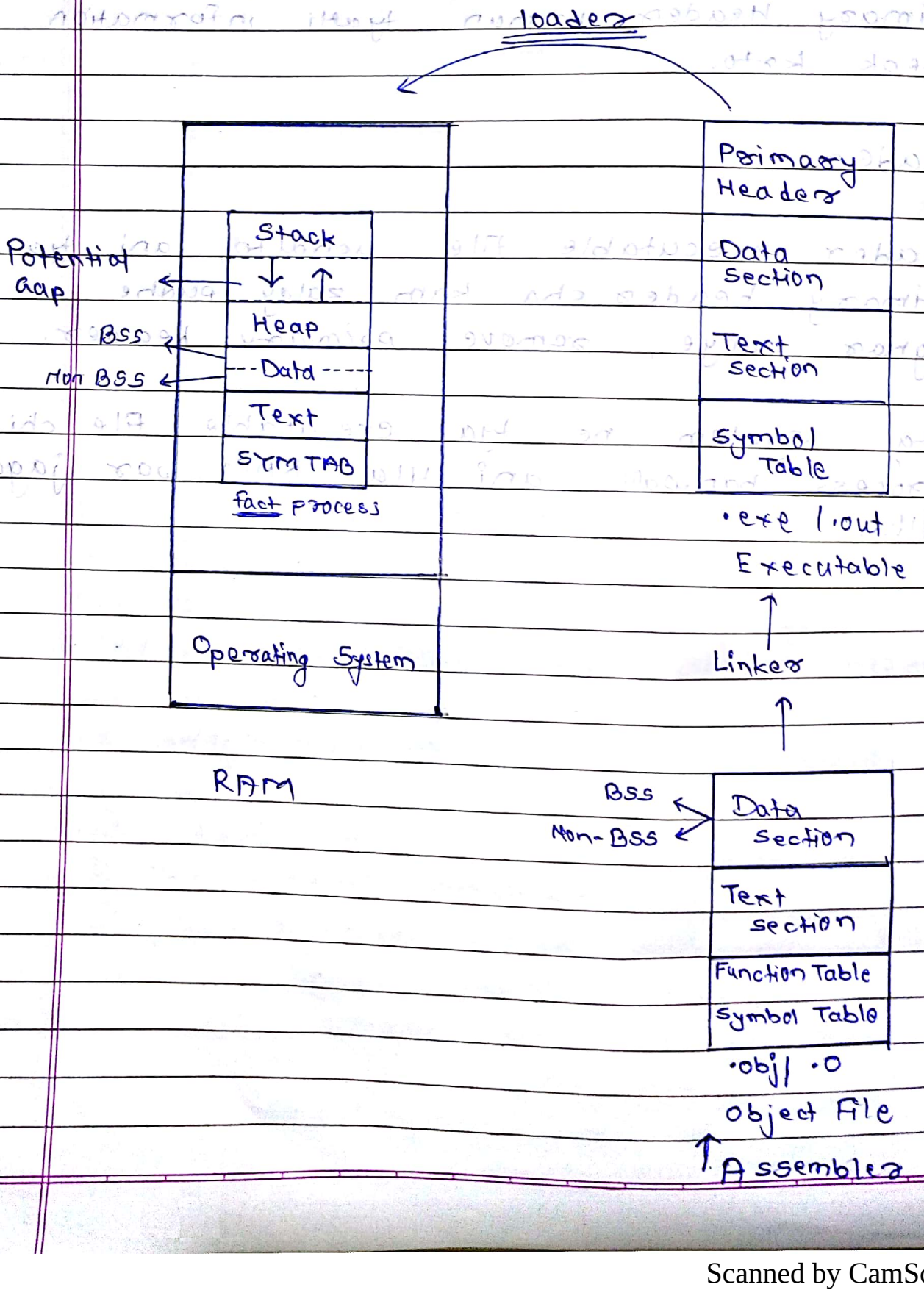
[ithun pudhe output dise paryant kay hotay te...]

Jeva apan executable run karto teva operating system cha loader yeto ani tya executablech Primary Header baghun tyatli information check karto.

5 Loader

Loader executable file uchalto, ani tya primary header cha kam zaloy pudhe lagnar naiye, remove primary header.

Ata system ne tya executable file chi process banvali ani tila RAM war jaga dilit...



- **Symbol table:**
Symtab madhe saglya variables chi information aste.

- **Function table:**
Ta table madhe saglya functions chi information aste.

- **Text section:**
Text section madhe functions definition astat mhanjech executable statement astat.

Ani he sagla Binary mdhe asta, machine readable language.

- **Data section:**
Ithe sagle global variable store astat.
Je variable global ahet ani initialise nai kelet te Bss part mdhe, te zeroed area asto
Je variable global ahet, ani initialise kelele ahet te non-Bss mdhe astat with tyanchi value.
∴ global variable by default zero astat.

- **Primary Header:**

Primary header is very important.
yat 4 gashti astat.

(i) Magic Number:

- Patek operating system cha magic no vegla asto.
- To number jar match zala tarch loader Executable File gheun jato.
- Windows - MZ

[open any .exe on windows in notepad++]

(ii) Address of entry point:

- Execution kuthun suru karycha tithla address.
- To Text section madhly entry point function cha address asto,
- apla main nai,
  aplya main chya adhipn functions astat.

(iii) Time - Date stamp:

- Information about executable.

(iv) Type of executable:

(a) Self executable
.exe

(b) Dependend executable
.dll

- Stack

- Jeva process run hote tevach stack banto, prtyek process la swtahacha stack asto.

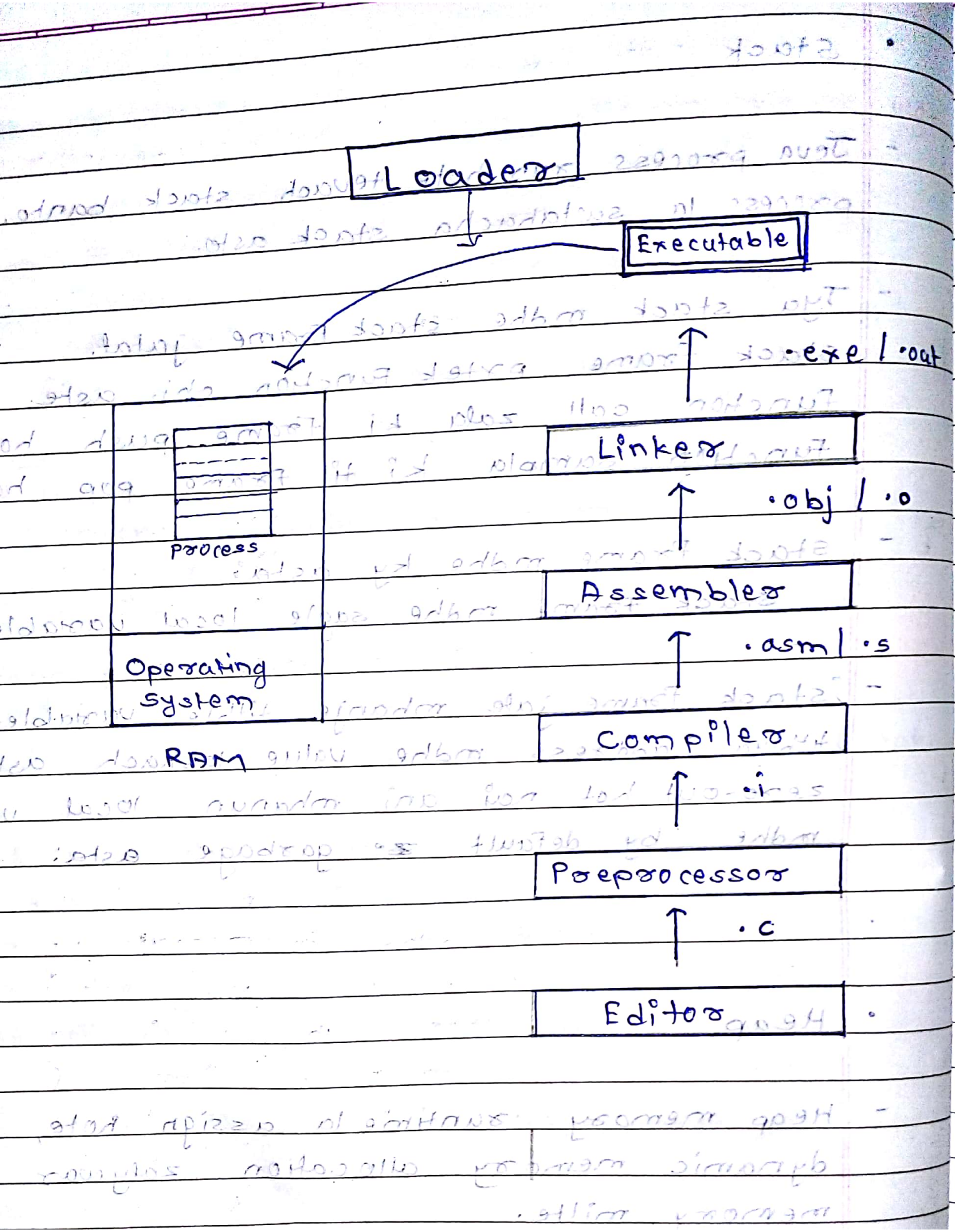
- Tya stack mdhe stack frame jatat.
  stack frame prtek function chi. aste.
  function call zala ki frame push hote,
  function sampla ki ti frame pop hote.

- stack frame mdhe ky asta?
  stack frame mdhe sagle local variable astat.

- stack frame jate mhanje tithle variables jatat tithlya address mdhe value taskach astat te zero-out hot nai ani mhanun local variable mdhe by default garbage asta.

- Heap

- Heap memory runtime la assign hote, i.e dynamic memory allocation zalyvar heapvar memory milte.



PATA = Integrated Drive Device Electronics (IDE) in HW

Integrated Development Environment (IDE) in SW

#

main function la 3 parameter astat,

prototype of main function:

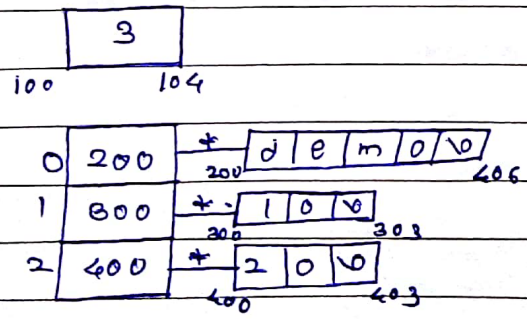
```
int main(
        int argc,
        char *argv[],
        char *envp[]
        );
```

argc : argument count, executable run kartana kiti parameter pass kele tyacha count, [include File name]

argv : 'argv' is array, which contains 'argc' element, each element is of type pointer, which points to 'character'.

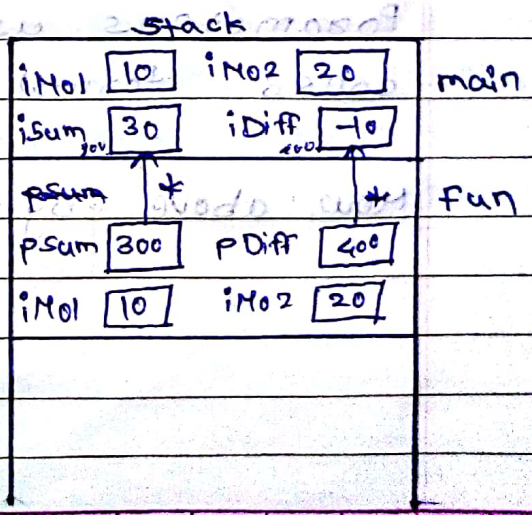
envp : same like 'argv', it store environmental settings. like path, etc... (given by parents process).

$ demo 10 20.



argv[0] = demo

```
① void fun(int, int, int *, int *);          // prototype / function def^n
② int main()
③ {
④     int iNo1, iNo2, isum, iDiff;
⑤     printf("Enter two numbers");
⑥     scanf("%d %d", &iNo1, &iNo2);
⑦     fun(iNo1, iNo2, &isum, &iDiff);        // Function call
⑧     printf("Addition is %d", isum);
⑨     printf("subtraction is %d", iDiff);
⑩     return 0;
⑪ }

⑫ void fun(int iNo1, int iNo2, int *psum, int *pDiff)
⑬ {
⑭     *psum = iNo1 + iNo2;
⑮     *pDiff = iNo1 - iNo2;
⑯ }
```



Explaination:

In above program we return multiple values from 'fun' indirectly using call by address.

Line 7: Here we passed 1st two parameter, using value (call by value), last two parameters using adder (call by address)

Line 12: Here we are getting 1st two parameter but we are not returning value from them, hence they are 'IN Parameters'

We are accepting address of variable in last two parameters where those variables (iSum, iDiff) doesn't any proper value but after function execution they contain proper values. Hence they are 'OUT Parameters'

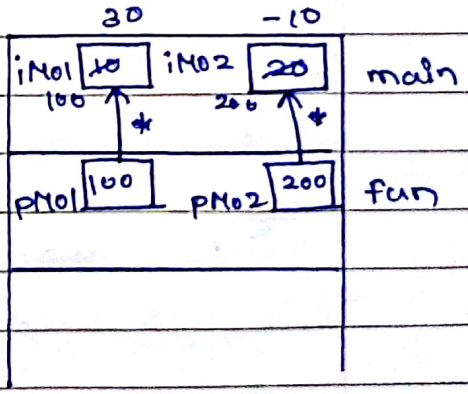
Note:

1) 'main' is 'caller' function and fun is 'callie' function.

2) Parameters use while calling function are called actual parameters.
Parameters use at function definition are called formal parameters.

Now, above by using IN-OUT parameter:

```
void fun(int *, int *);

int main()
{
    int iNo1, iNo2;

    printf("Enter two numbers");
    scanf("%d%d", &iNo1, &iNo2);

    fun(&iNo1, &iNo2);

    printf("Addition is %d", iNo1);
    printf("Subtraction is %d", iNo2);

    return 0;
}

void fun(int *pNo1, int *pNo2)
{
    *pNo1 = *pNo1 + *pNo2;
    *pNo2 = *pNo2 - *pNo1;
}
```



return 0	: Successfull
return 1	: Abortive
return -1	: Erroneous

```c
#include <stdio.h>
int fact (int);
int main()
{
    int iNo, iAns;
    printf("Enter number");
    scanf("%d", &iNo);
    iAns = fact(iNo);
    printf("Factorial is %d", iAns);
    return 0;
}

int fact(int iNo)
{
    if (1 == iNo)
        return 1;
    else
        return iNo * fact(iNo - 1);
}
```

iAns = fact(5)

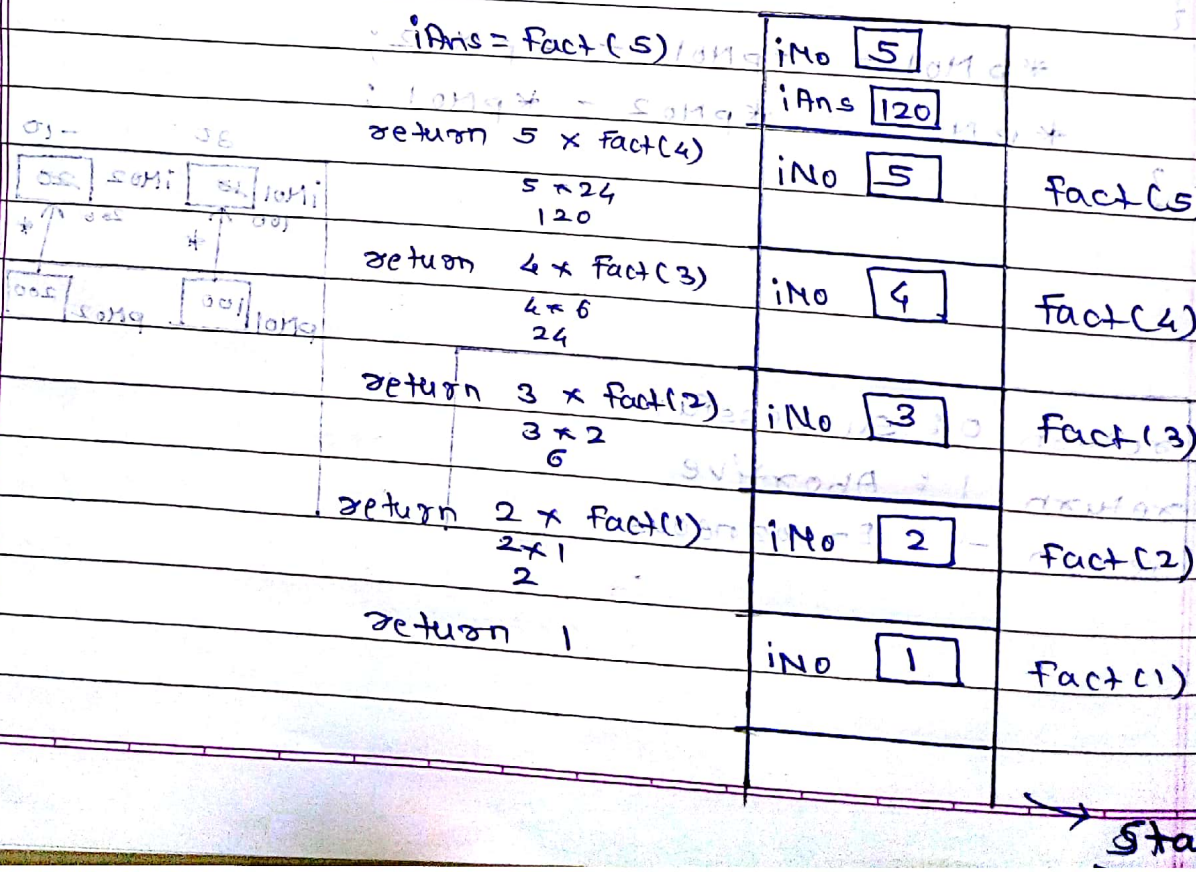
return 5 * fact(4)
5 * 24
120

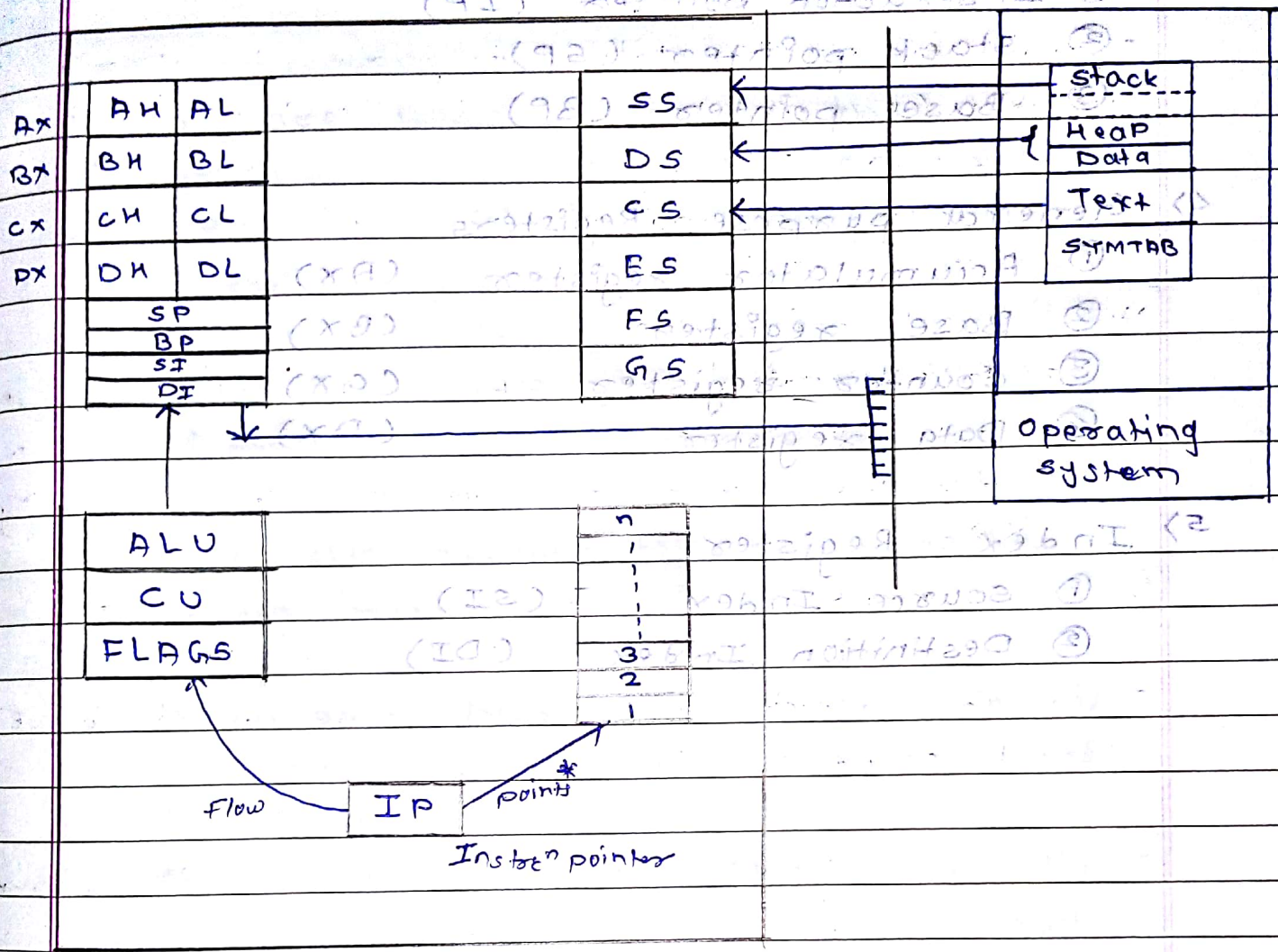
return 4 * fact(3)
4 * 6
24

return 3 * fact(2)
3 * 2
6

return 2 * fact(1)
2 * 1
2

return 1





1) Segment Registers
   1. Stack segment (SS)
   2. Data segment (DS)
   3. Counter segment (CS)
   4. Extra segment (ES)
   5. FS
   6. GS

2) Flags Register.

3) Pointer Registers
   1. Instruction pointer (IP)
   2. Stack pointer (SP)
   3. Base pointer (BP)

4) General purpose Registers
   1. Accumulator register (AX)
   2. Base register (BX)
   3. Counter register (CX)
   4. Data register (DX)

5) Index Register
   1. Source Index (SI)
   2. Destinition Index (DI)