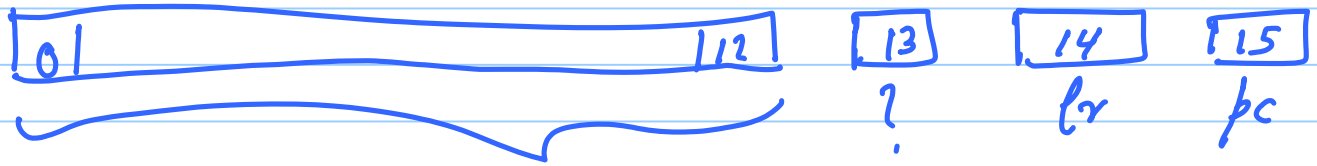
Aug 1st

Note Title 01-08-2012

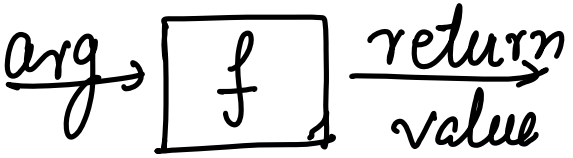


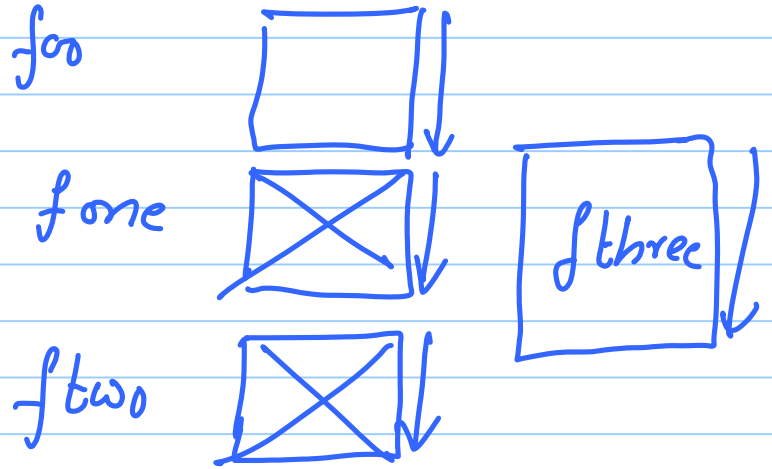
X Need more variables.

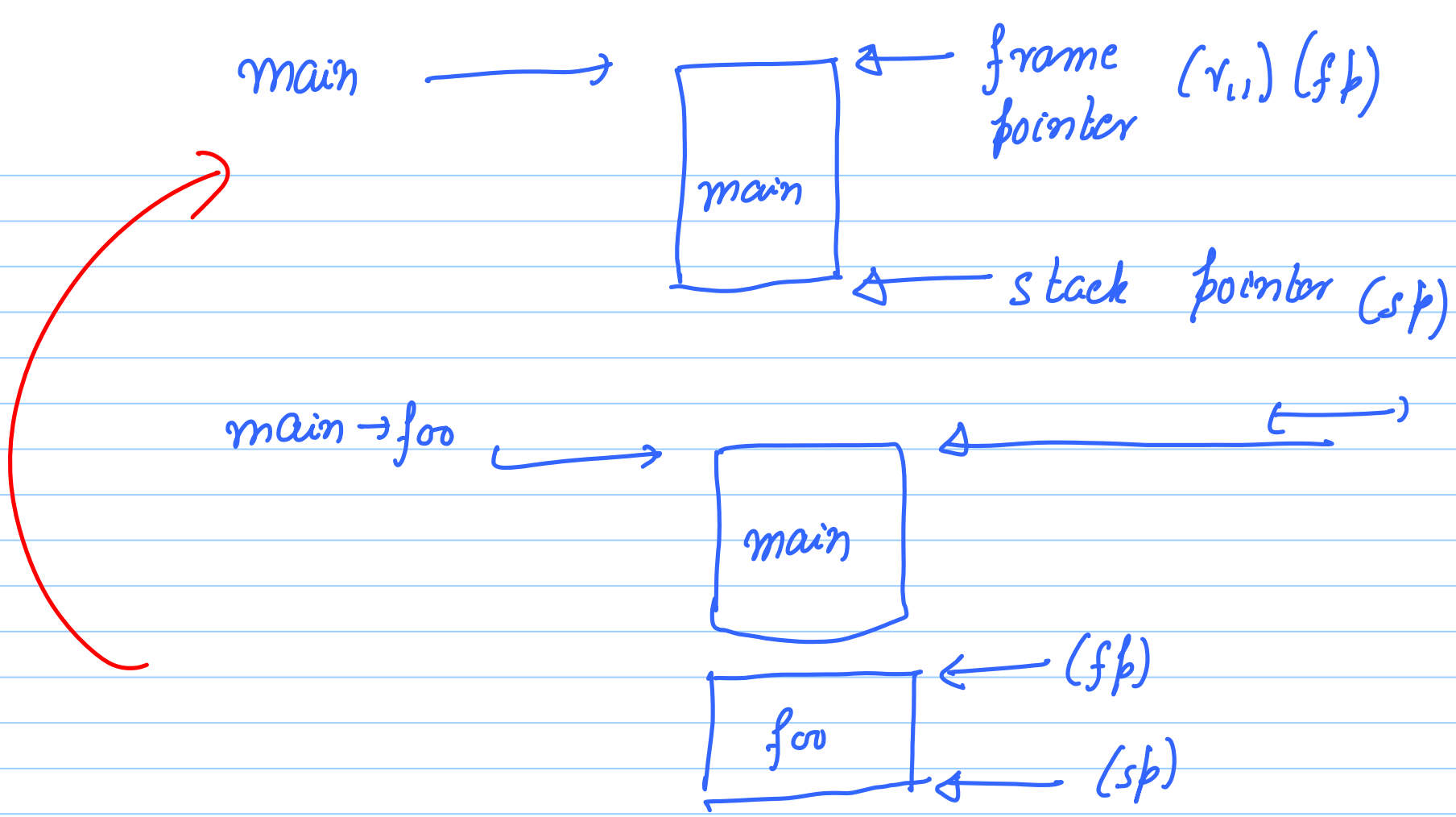
foo → fone → ftwo → { int a, b, c; }

```
int ftwo(){
    int a, b, c;
    .
    .
    return (a+b-10);
}

foo {
    fone();
    fthree();
}

fone {
    ftwo();
}
```

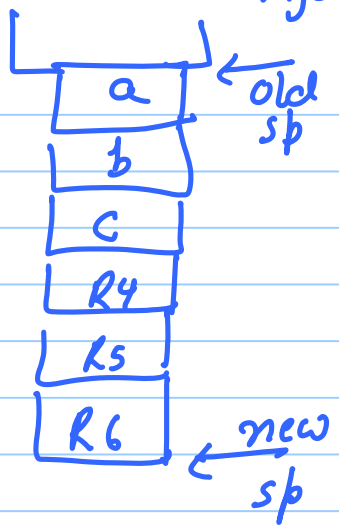
automatic vars.






main
frame pointer (fp)
main
stack pointer (sp)
main → foo
main
(fp)
foo
(sp)

```
foo() {
    int a, b, c;
```



main → foo
(caller) (callee)

✓ more than 13 variables

✓ issue of recursion.

(callee saved)

```
ADD  sp, sp, #-24
STR  R6, [sp, #0]
STR  R5, [sp, #4]
STR  R4, [sp, #8]
 :
MOV  R6, #10
 :
LDR  R6, [sp, #0]
LDR  R5, [sp, #4]
LDR  R4, [sp, #8]
```

save

restore

reclaim space →

```
ADD  sp, sp, #24
```

restore sp

Compilation Process

1) a.c gcc a.c ./a.out

1) www.gnuarm.com ✓ (download & install)
   arm-elf-gcc
   arm-elf-run
2) palasi (details to be provided)
3) emuArm [end of this week or next week]

arm-elf-gcc -S ab.c

[course webpage]

ab.s

Assembly ?

C → Assembly

Barring a few exceptions

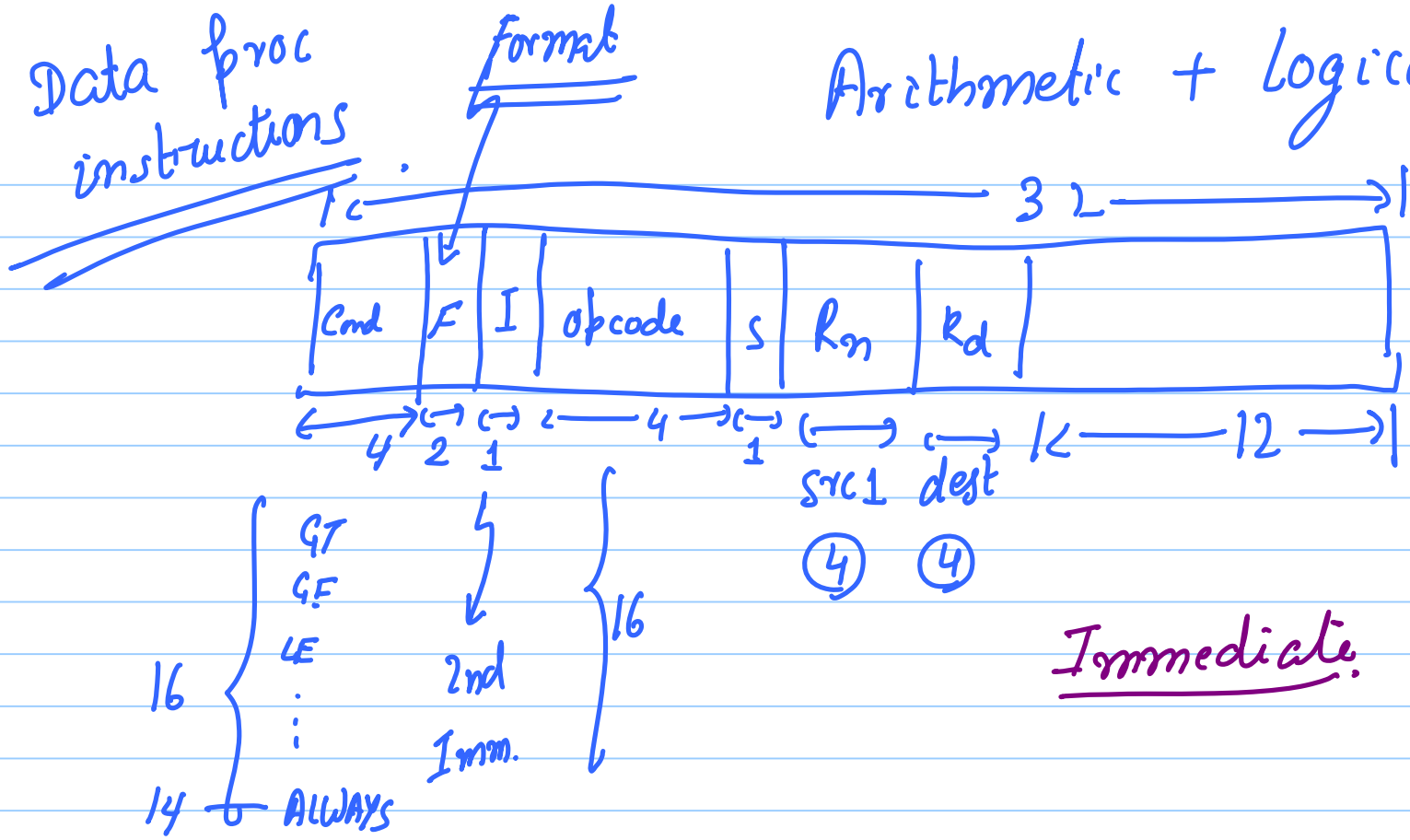
1 assembly statement ⟷ 1 machine instruction

Convert : 1 assembly statement to sequence of 32 bits

Data proc instructions

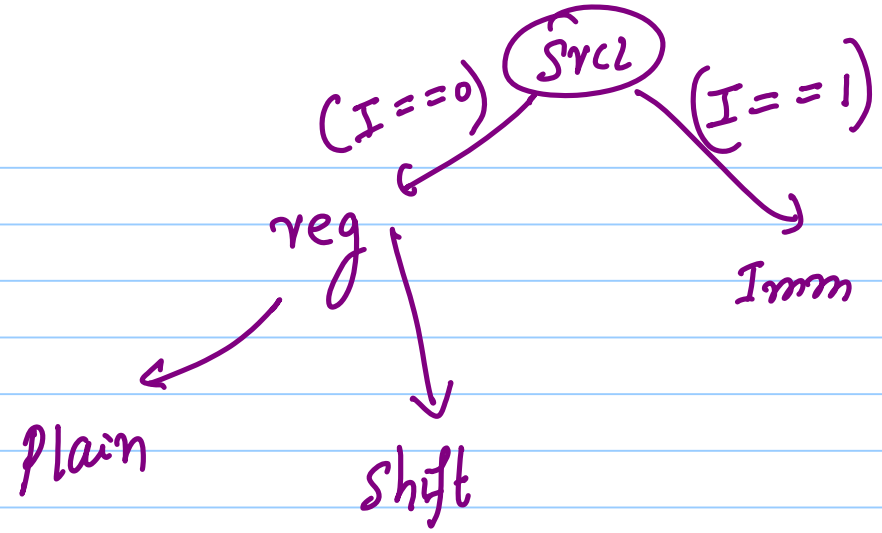
Format

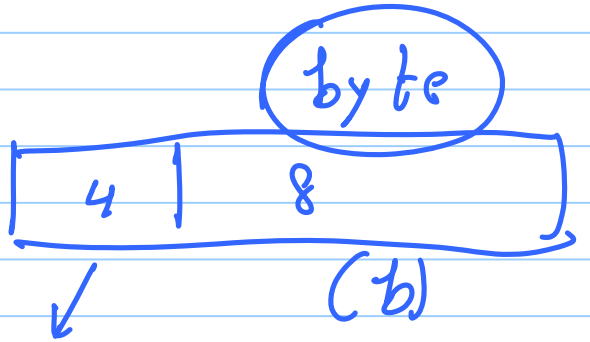
Arithmetic + Logical. (F = 00)



Immediate.

32
-10
22
-8
14
-2
12





0-15

$Imm = b \text{ ROR } r$ $\quad \times 2$

0-30 (even)

r

E.g. $b = 0xFE$

E0 00 00 0F

00. FE 00 00

$I_1 = 0x\ FE\ 12\ 39\ A7$

```
MOV   R0, #A7
ORR   R0, R0, #3900
ORR   R0, R0, #120000
ORR   R0, R0, #FE000000
```

(b) (hex)

1100 | 39

4 | 8