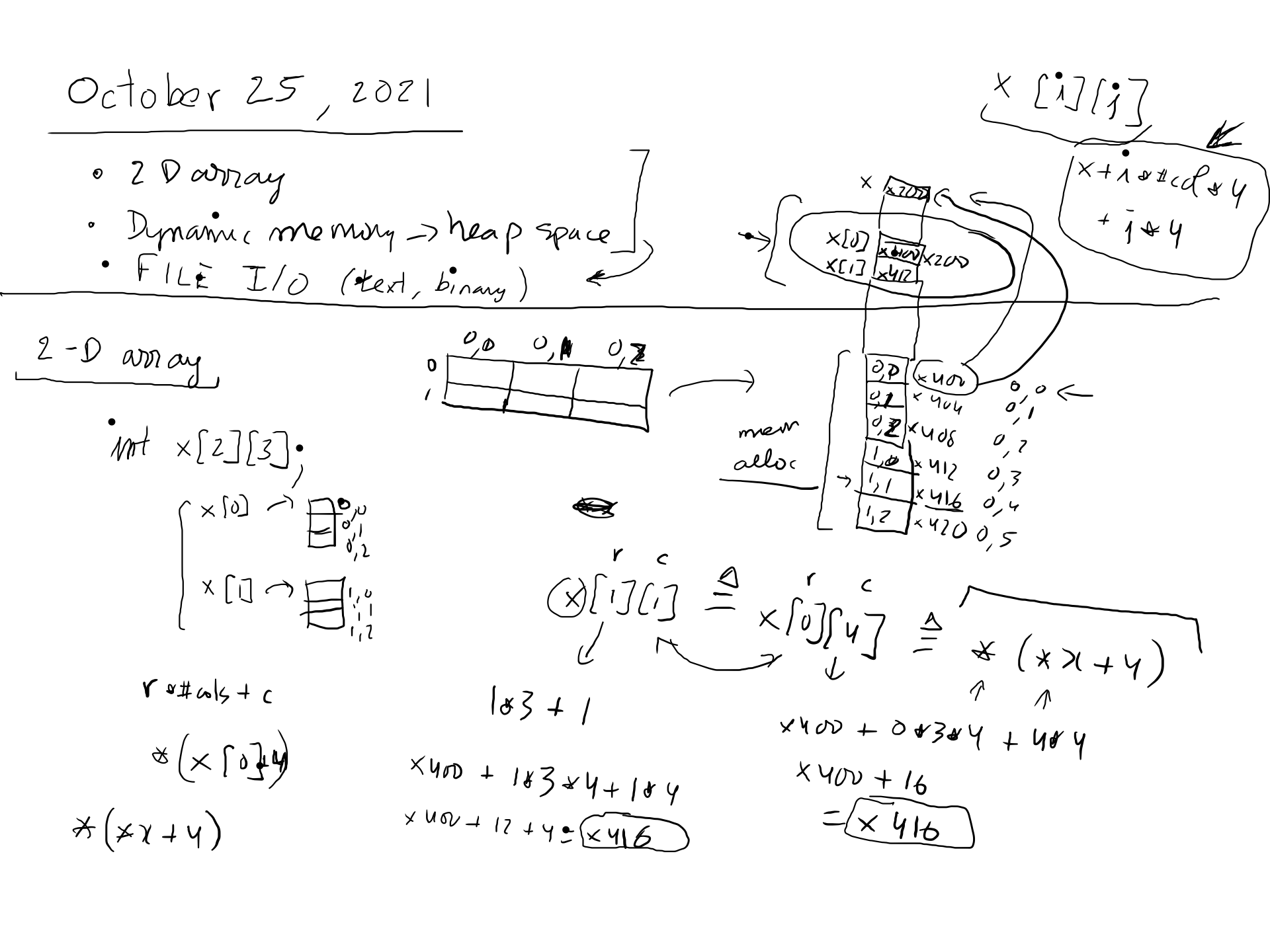
October 25, 2021

- 2D array
- Dynamic memory -> heap space
- FILE I/O (text, binary)



2-D array

- `int x[2][3];`

r * #cols + c

*(x[0] + 4)

*(*x + 4)

x[1][1] ≡ x[0][4] ≡ *(*x + 4)

1 * 3 + 1

x400 + 1 * 3 * 4 + 1 * 4

x400 + 12 + 4 = x416

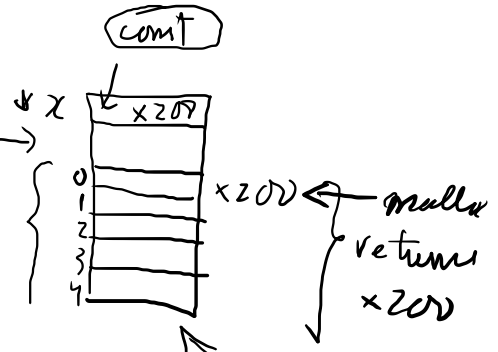
x400 + 0 * 3 * 4 + 4 * 4

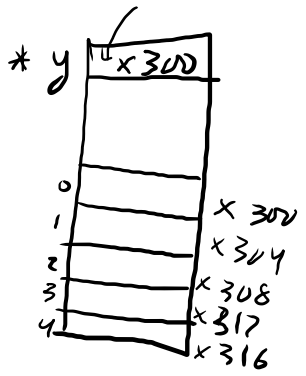
x400 + 16

= x416

Dynamic Memory

```
int x[5];
int y;
x = &y;
```





malloc – memory allocation

```
(int *)malloc(#bytes to allocate)
```

address of 1st memory of the block of mem.

```
int x[5];  ⟶  int *y;
```

```
int a;
```

#of mem. elts

```
y = (int *)malloc(5 * sizeof(int))
```

x300

```
y = &a;
```

using in ARM to alloc. mem.

`int x[5];` 5 * 4 = 20

5 * sizeof(int)

// x = (int *)malloc(#bytes);

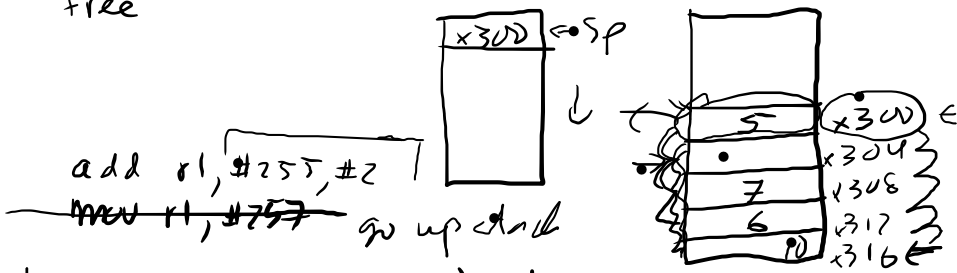
```
mov r0,#5
mov r1,#4
mul r0,r0,r1 @ 5*4 → ALU
@ mov r0,r0,LSL #2   @ x4
```

`ldr r6, [r0,#16]` ← x[4]

delete [] x; r5 ← malloc(20)

free

x300 stack



bl malloc @ return address of 1st mem loc.

```
@ mov r5,r0
push {r0}
:
:
:
pop {r0} @ temp put addr x300 → r0
mov r1,#5
str r1, [r0] @ *r0 = r1
```

```
mul r0,r0,#4
mul r0,r0,r1
div
```

Largest immediate is 255

```
mov r1,#257
```

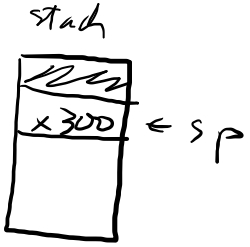
```
mov r1, #6        3*4
                  cannot be an immediate
str r1, [r0, #12]
mov r1, #7
str r1, [r6, #8]
```

`int x[2][3];`

1. Calculate # bytes = #rows * #cols * sizeof(int) → r0

2. call malloc

3. a) store r0 into another reg (r4 → r10); b) push on the stack

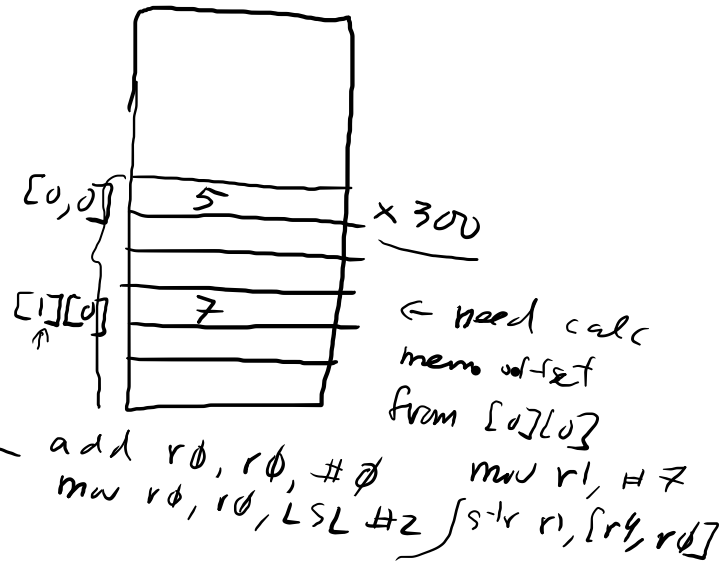



```
mov r0, #2
mov r1, #3
mul r0, r0, r3  @ r0 = 2*3
mov r0, r0, LSL #2  @ r0 = 4*r0
```

↓ bytes of an int

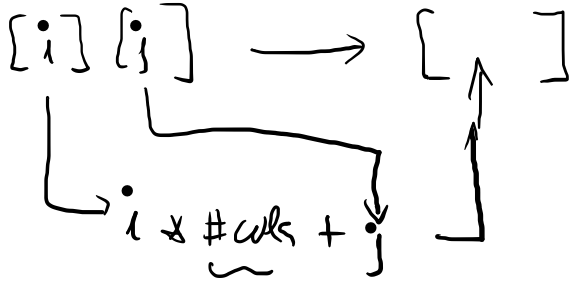
```
bl malloc  @ → r0
push {r0}
mov r1, #5
pop {r4}
str r1, [r4]
```

```
mov r0, #1
mov r1, #3  @ #cols
mul r0, r0, r1  @ i * #cols
```




```
add r0, r0, #0
mov r0, r0, LSL #2
```

```
mov r1, #7
str r1, [r4, r0]
```



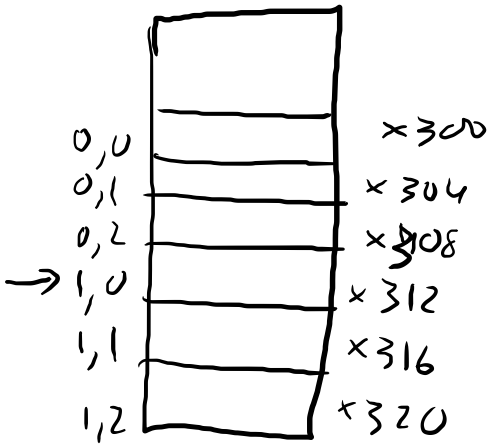
#cols = 3

rows

x [2][3]

cols

heap



[1][0]

cols

bytes of int

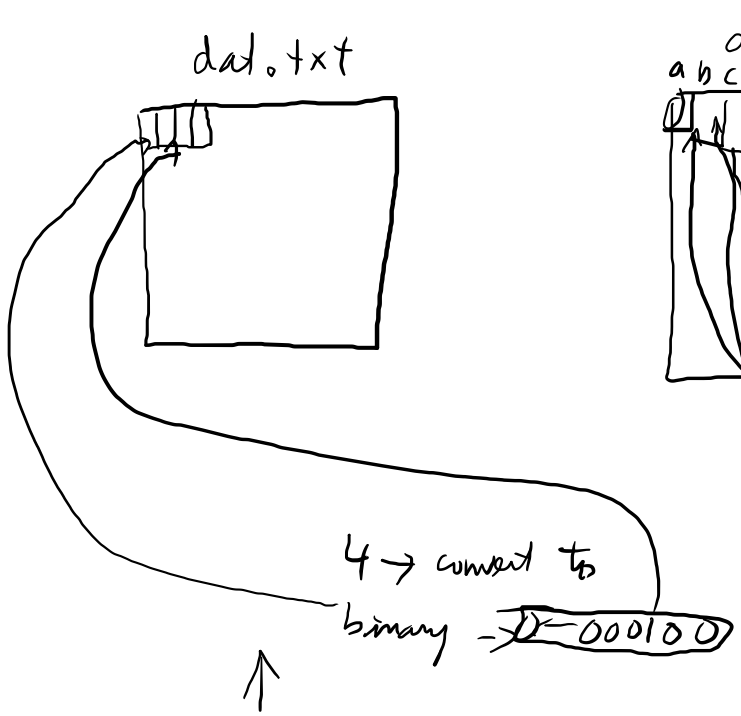
offset = (1 * 3 + 0) * 4 = 12

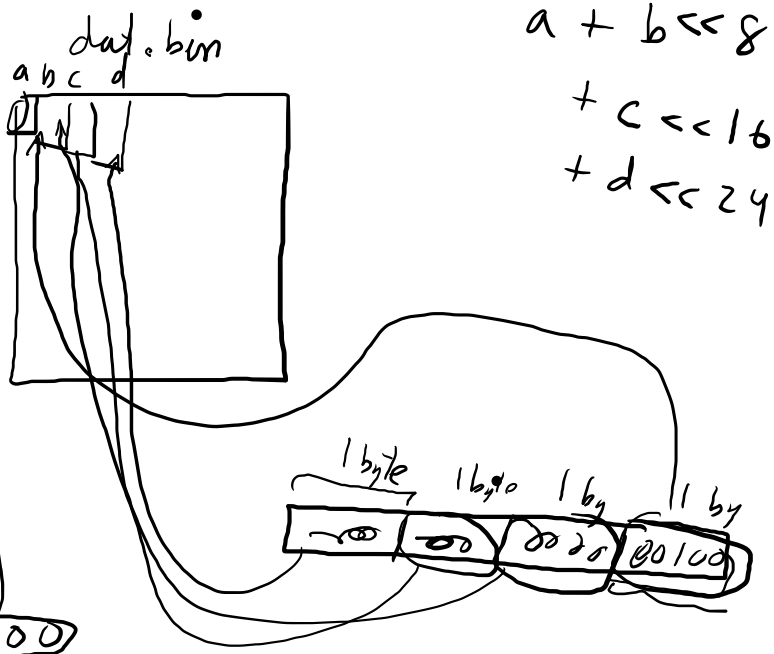
[x300, #12]

addr of 1st elt

offset to [i][j]

File I/O → Text File

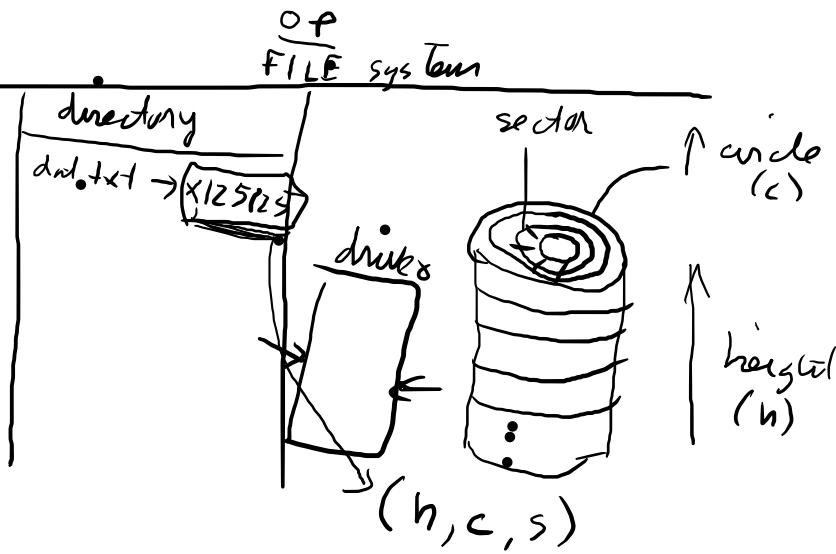





```
a + b<<8
  + c<<16
  + d<<24
```

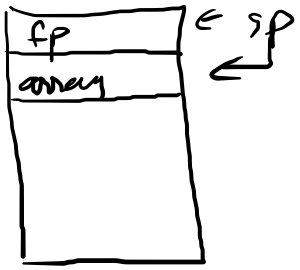
c

```
FILE *fp;
fp = fopen("dat.txt", "r");
fscanf(fp, "%d", &x);
fclose(fp);
```



FILE I/O in ARM

```
.data
filename:   .asciz   "dat.txt"
readopt:    .asciz   "r"
inp:        .acsiz   "%d"
    .
    .
```



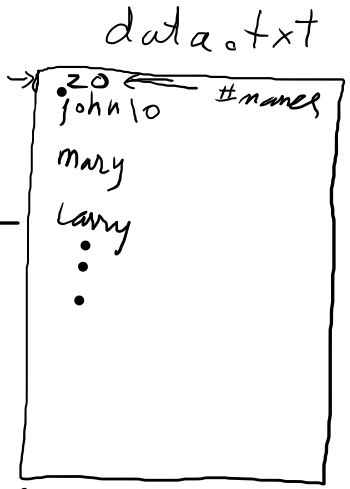

```
                    r0          r1
@ fp = fopen( filename, "r" );

ldr r0, =filename

ldr r1, =readopt

bl fopen @ r0 contains file address ≙ fp

push {r0}
```

@ fscanf (fp, "%d", &x)
r0 r1 r2

```
ldr r0, [sp]  @ r0 = fp
ldr r1, =inp
sub sp, sp, #4
mov r2, sp
bl fscanf
add sp, sp, #4
pop {r0}  @ get r0 = fp
bl fclose  @ fclose (fp)
```

Ex: (2-D, dyn mem, FILE I/O, char)

Read in a data file containing a bunch of names, print out the names!



main

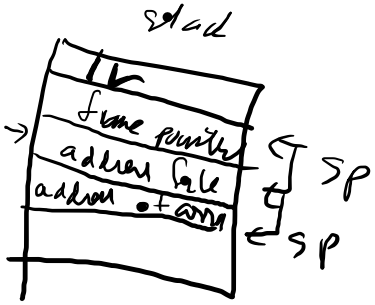
- open a file for reading → call fopen → address of file
  push {r0}
1. Read in metadata → # names
2. Calc. total # of bytes → # * 80 * sizeof(char) (1)
3. Allocate memory using malloc
   r0 ← malloc (total # bytes)
   push {r0}
4. loop 0 → #names-1
   assign read char 1 at a time until end of line
5. print out names

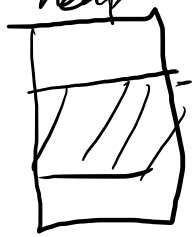
char **x;
char [20][80]

20 → a

20 x 80 (1)

x = (char *) malloc (a * 80 * sizeof(char))





main.s

```
.cpu cortex-a53
.fpu neon-fp-armv8

.data
filename: .asciz "data.txt"
readopt: .asciz "r"


.text
.align 2
.global main
.type main, %function

main:
    push {fp, lr}
    add fp, sp, #4
```

@ open the file for reading

@ fp = fopen("data.txt", "r"); ← file ptr
r0 r1

ldr r0, =filename

ldr r1, =readopt

bl fopen @ fp → r0

push {r0}

@ get the #names from file metadata

@ readmeta(file ptr) → r0 = # of names
r0

ldr r0, [sp] @ r0 = file ptr

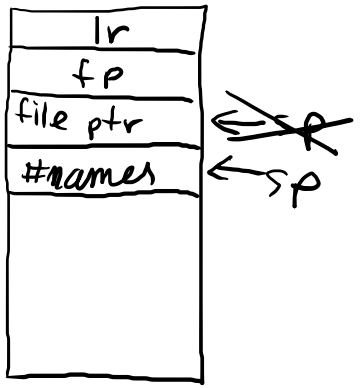
bl readmeta @ #names → r0

push {r0}

@ allocate memory = #names * 80 + 1

ldr r0, [sp]
mov r1, #80
mul r0, r0, r1

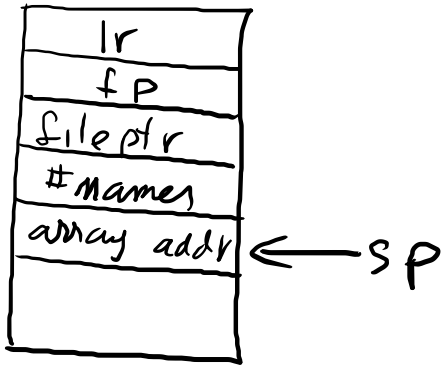
stack



fp - file ptr
- frame ptr

bl malloc @ malloc(r0)

push {r0}



.
.
.

sub sp, fp, #4

pop {fp, pc}