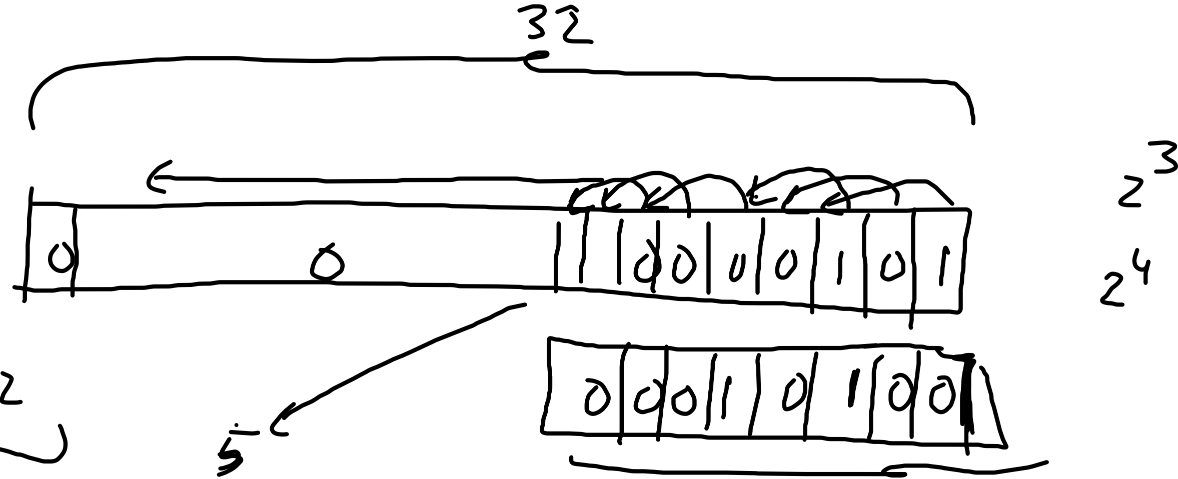
October 18, 2021

LSL $\triangleq$ logical shift left LSL #n ↑

LSR $\triangleq$ logical shift right

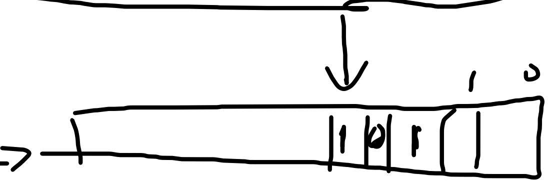
LSL #2 shifts bits over by 2 bits to the left

mov r5, #5



mov r5, $\underbrace{\#5 \quad LSL \#2}$

$$\left[- \cdots 0 + 0 + 1\times 2^{\overset{4}{\cancel{2}}} + 0\times 2^{\overset{3}{\cancel{1}}} + 1\times 2^{\overset{2}{\cancel{0}}} \right] \times 2^2$$



2D arrays

col 1	col 2	col 3	
0,0	0,1	0,2	row 1
1,0	1,1	1,2	row 2

$[1][1] \rightarrow 1 * \boxed{3} + 1 = \boxed{4}$

int x[2][3];

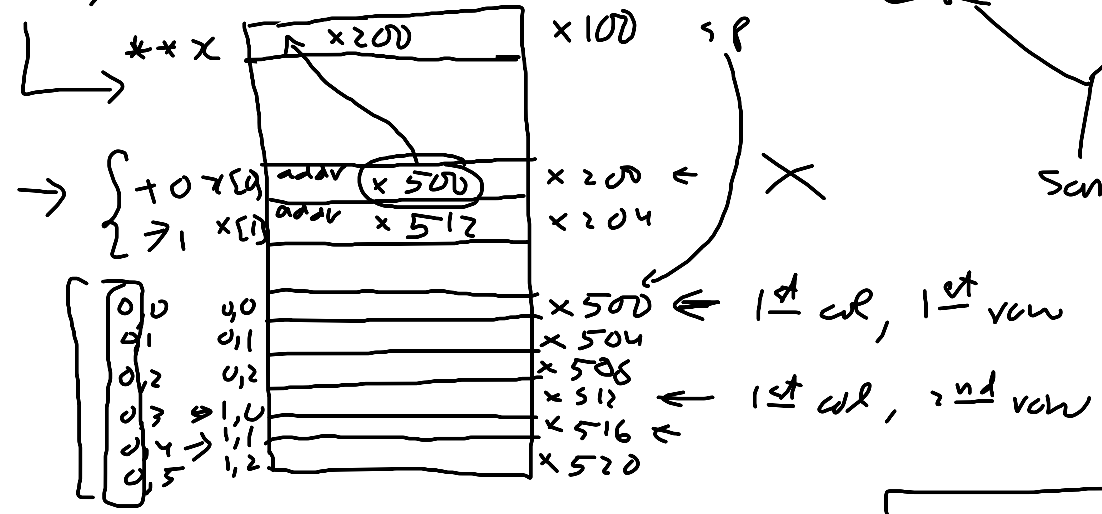
$[1][0] \rightarrow 1*3+0$
$= 3$

test(x[][3])

$\underbrace{x[0][4]}_{} \triangleq \underbrace{x[1][1]}_{}$

Mapped
Same location

$\boxed{X}$



$(x[0])[4]$

$(x500 + 4*4) = \underline{x516}$

↓

$[0][\boxed{4}] \leftrightarrow \underbrace{[1][1]}_{}$

[4]

[5]

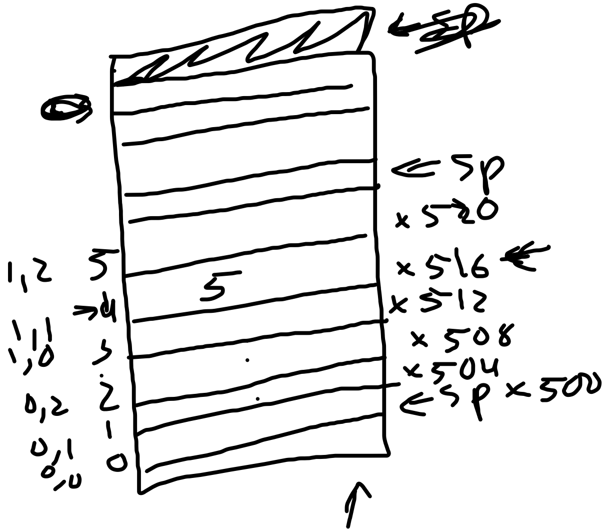
$(x[0])[0] \rightarrow (x500)[0] \rightarrow (x500 + 0*4)$
$= x500$

$x[0][1] \rightarrow (x500)[1] \rightarrow (x500 + 1*4)$
$= x504$

$x[1][1] \rightarrow (x[1])[1]$
$(x512)[1] \rightarrow (x512 + 1*4)$
$\boxed{x516}$

$x\underbrace{[i][j]}_{o} \longrightarrow x[0][i * COLS + j]$

$[i][j] \longrightarrow i * COLS + j$



int x[2][3]

$2*3*\overset{\text{bytes}}{4} = 24$ bytes

```
sub sp, sp, #24
```

$x[0][0] \rightarrow (0 * \overset{\text{COLS}}{3} + 0)^{*4\ \text{bytes}} = 0 * 4 = 0$

```
mov r0, #1
str r0, [sp, #0]  @ x[0][0] = 1
```

$x[i][j] \longrightarrow$ mem offset $= (i * COLS + j) * 4$ bytes

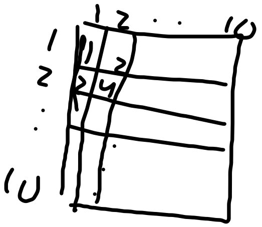
```
str r0, [sp, offset]
```

$x[1][1] \rightarrow (1 * \overset{\text{COLS}}{3} + 1) * 4$

$= 16$ bytes

```
mov r0, #5
str r0, [sp, #16]
```

Ex: prog. generate a mult. table 10 x 10



main.s

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

.data

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

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

```
@ allocate memory
mov r0, 10
mul r0, r0, r0
mov r0, r0, LSL #2
sub sp, sp, r0  @ allocates 10x10 mem locs
@ r10=i | r9=j
mov r10, #0
out_loop:
   cmp r10, #10
   bge end_out_loop

   mov r9, #0
  in_loop:
     cmp r9, #10
     bge end_in_loop
```

```
for (i=~~1~~0; i≤10; i++)
  for (j=~~1~~0; j≤10; j++)
    x[i-1][j-1] = i*j;

    x[i][j]

(i*10+j)*4
       ^ mem
         offset
```

```
@ a[i][j] = (i+1) * (j+1)
@ offset = (i * 10 + j) * 4
@ = (r10 * 10 + r9) * 4
mov r2, #10
mul r2, r10, r2     @ r10 * 10
add r2, r2, r9      @ r10*10 + r9
mov r2, r2, LSL #2  @ r2 = 4 * (r10*10 + r9)
@ (r10+1) * (r9+1)
add r3, r10, #1     @ r3 = r10+1
mul r4, r3, r9      @ r4 = (r10+1) * r9
add r3, r3, r4      @ r3 = r4 + r10+1
str r3, [sp, r2]
add r9, r9, #1
b in_loop
```

```
end_in_loop:
      add r10, r10, #1
      b out_loop
end_out_loop:
      @ printArray
      mov r0, sp
      mov r1, #100
      bl printArray

   sub sp, fp, #4
   pop {fp, pc}
```