Note Title 04-08-2012

Aug 4th

Assembly. ADD, SUB, ORR, AND, EOR

format : [<opcode> dest src1 src2]

variables : R_0, - - - - - - R_{12}

✓ 2s complement

$\text{ADD} \xrightarrow{\text{2 formats}}$

ADD $R3, R2, R1$ $\quad R_3 = R_1 + R_2$

ADD $R_3, R_2, \#4$ $\quad R_3 = 4 + R_2$

$\left[\begin{array}{l} \text{LSL} \\ \\ \text{LSR} \\ \text{ASR} \end{array}\right]$

R_3, R_1, R_2 $\quad R_3 = R_1 << R_2$

$R_3, R_1, \#2$ $\quad = R_1 << 2$

Src 2 can be an immediate

✗ not src 1

What else do I require?

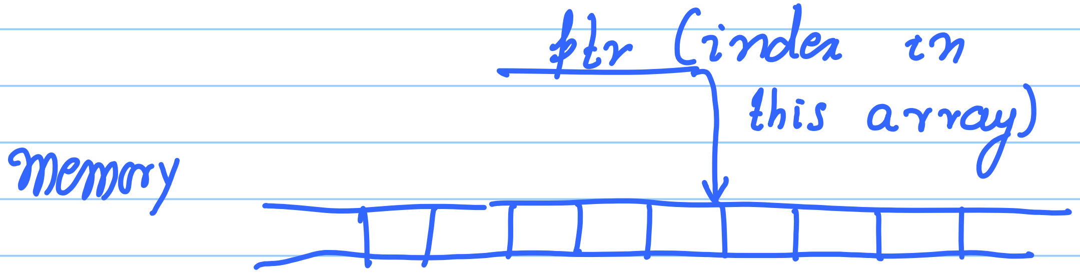
✓ if-else ✓ arrays.

✓ loops functions.

conditionals

Extra register → R_{15} (PC)

(Program Counter)



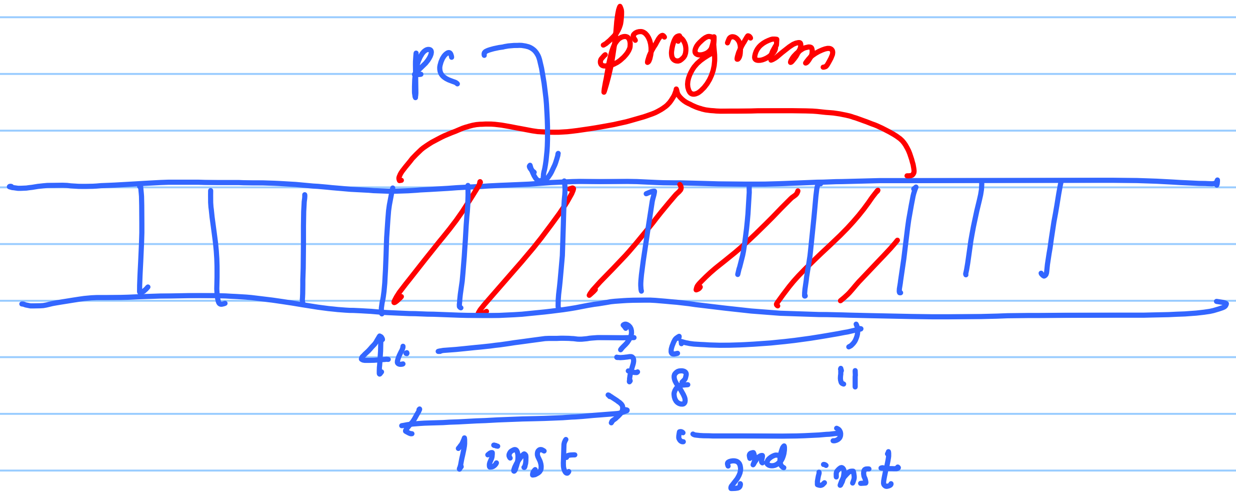
one large byte array.

PC is a pointer (special)

Entire program: Stored as an array of instructions.

1 assembly line ⟷ 1 machine instruction.

[32 bits]
{4 bytes}



pc (ptr) to the memory segment holding the program

```
                                 🚩mov   R0, #5     (a=5)
.L1
  ADD R2, R0, R1                   mov   R1, #5
   🚩
  B   .exit                      🚩CMP   R0, R1

                                 🚩BEQ   .L1

                               .exit
```

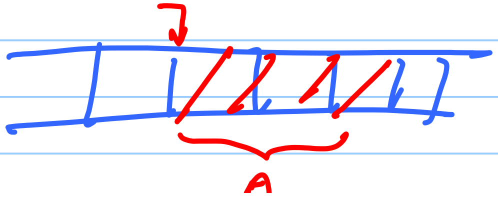
```
R0          R1
a = 5;      b = 5;

if (a == b) {
     R2 c = a + b;
}
```

```
sum=0;
for (i = 0; i<100; i++) {
        sum += i;
}
```

Arrays.

```
int   A[10];
for (i=0; i<10; i++)
        A[i]= i;
```

Array : A
region of 40 bytes
in mem



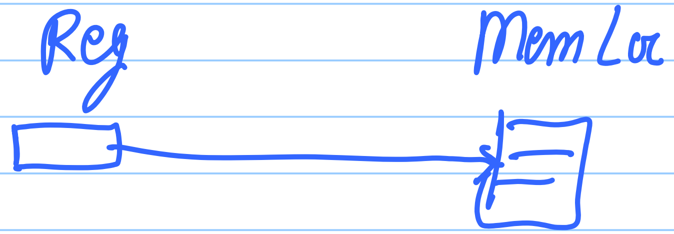
New instruction: Store (STR)

format: $\text{STR} \ \underbrace{R_4}_{\text{reg}}, \ \underbrace{[R_1, \#10]}_{\text{memory loc.}}$

$$mem(R_1+10) = R_4$$

memory location: $[\text{base\_reg}, \overset{[\text{reg}, \text{imm}]}{\text{offset}}]$

$$\text{memloc} = \text{value}(\text{base\_reg}) + \text{offset}$$

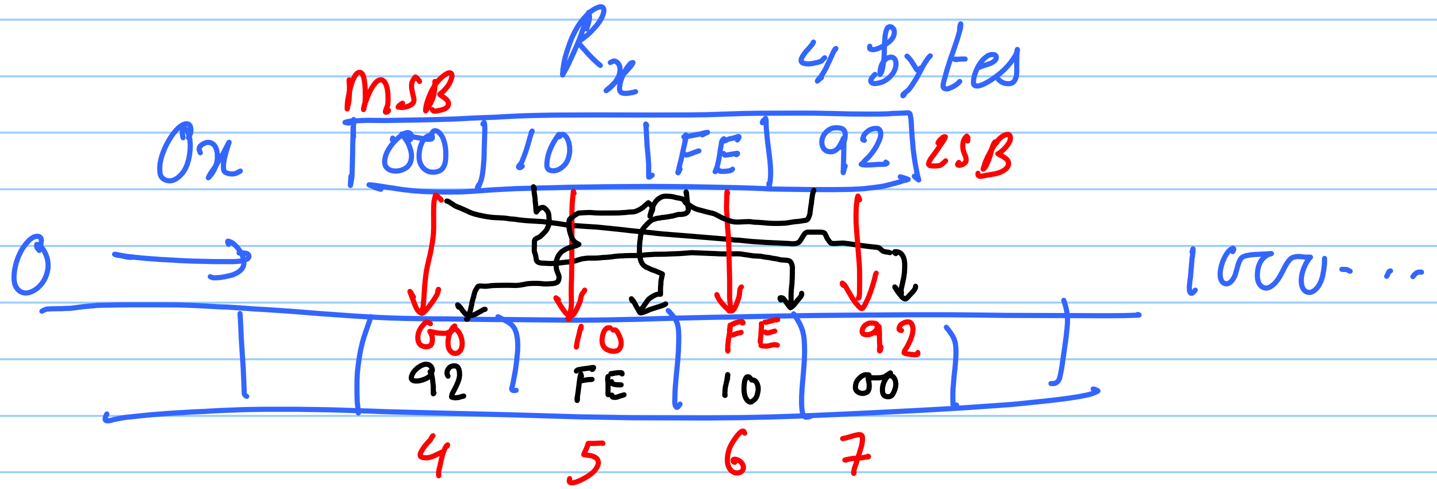


Every register has a storage space equal to
4 bytes
= 32 bits

STR → Transfer 4 bytes from a reg.
to memory

Variants of STR
STRH (2 bytes)
STRB (1 byte)

Register



(Big Endian) { IBM, HP
JAVA

(Little Endian) { ARM
INTEL, AMD

```
{ int   A[10];
  for (i=0; i<10; i++)
          A[i]= i;
}
```

$\underset{R_4}{\boxed{x}} = \overset{R_1}{A}[5];$)) ⟶

$$\begin{tikzcd} \text{reg} & \text{mem} \arrow[l, bend right] \end{tikzcd}$$

```
{ LDR    R4, [R1,#20]
  LDRH
  LDRB
```

ADD R_{dest}, R_{src1}, $\underset{src2}{\boxed{}}$

LDR R_{dest}, [R_{src1}, $\underset{src\ 2}{\boxed{}}$]

Src 2 :

immediate :	#10
register :	r_{src2}
imm. scaled reg :	r_{src2}, LSL #10
reg. scaled reg :	r_{src2}, LSL r_{src3}

{ B .label

{ B #offset newPC = oldPC + offset + 8