COL216 Lecture 1

Rishabh Dhiman

6 January 2022

We look at ARM (Advanced RISC Machine).

- 32-bit instruction set
- 16 registers

```
a = b + c

add r1, r2, r3
```

Since there are 16 registers, we require 4 bits to specify them. Consider, the add instruction what do we do with the remaining $20 = 32 - 3 \times 4$ bits, use all of it to specify add?

There are 2^{32} bytes, numbered $0, 1, \ldots, 2^{32}-1$, and 2^{30} aligned words, $0, 4, \ldots, 2^{32}-4$. There exist other misaligned words too, what happens to them?

```
v = a[8]

ldr r1, [r2, #32] // r2 corresponds to the a array

a[8] = a[8] + h

ldr r1, [r2, #32]
add r1, r1, r3 // r3 corresponds to h
str r1, [r2, #32]
```

ldr loads into the left register from the right register. str loads into the right register from the left register. No wait, are both of them registers or one of them must be memory?

// I missed swap waala

1 COntrol Flow

```
if (i == j)
        h = i + j;
k = k - i;
```

```
cmp r1, r2
bne L // branch if not equal
add r3, r1, r2
L: sub r4, r4, r1

if (i == j)
        h = i + j;
else
        h = i - j;
k = k - i;

cmp r1, r2
bne Lab1
add r3, r1, r2
b Lab2 // unconditional branch
Lab1: sub r3, r1, r2
Lab2: sub r4, r1
```

Similarly we have blt, stuff

Summation:

```
s = 0
i = 0
L: s = s + A[i];
i++;
if (i < n) goto L;

mov r1, #0
mov r2, #0
L: mul r3, r2, r7 // r7 = 4
ldr r4, [, r3] // I fucked up
add r3, r3, r4 // r3 corresponds to s
add r2, r2, #1
cmp r2, r6 // r6 = n
blt L
```