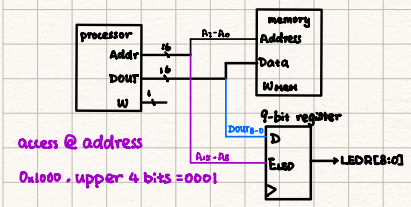

- Want a program to take the values on the inputs switches - 9 of them & then puts that "value" on to the 9 LEDs on DE1-SoC → part 3
- assume that the LEDs are "memory-mapped" to address 0x1000

switches → 0x3000

- Assembly-language Program: use "assembler" to translate it into a sequence of "machine code" numbers.

```
.define  LED_ADDRESS   0x1000     not executed.
.define  SW_ADDRESS    0x3000     just makes program easier to write & read for humans

        mvt     r3, #LED_ADDRESS     r3 ← 0x1000   0 0011 011 0000 10000
        mvt     r4, #SW_ADDRESS      r4 ← 0x3000
MAIN:   ld      r0, [r4]             r0 ← data from address in r4 = 0x3000
        st      r0, [r3]             address in r3 (LEDs) ← r0
        mv  r1, #2  or  mv  PC, #2       PC = r1 ← 2 . Sets the program counter to 2 → next instruction to be executed will be fetched from address 2
    or  mv  PC, #MAIN or b  #MAIN        creates an infinite loop of 2 load & store instruction
```

- there are both unconditional branches (e.g b #MAIN)
- & conditional branches → do the branch only if some condition is true

example: the value in G register is zero


```
        beq  #MAIN    go to #MAIN if z=1
        mv   r1, r2   other wise, just go to next instruction
        bne  #MAIN    go to #MAIN if z=0
        bcc  #MAIN    go to #MAIN if c=0
        bcs  #MAIN    go to #MAIN if c=1
```

Encoding is different: bicond't #D

cond: 111 1 xxx DDDDDDDDD

condition: 000 none, 001 eq, 010 ne, 011 cc, 100 cs

- Relates to part 6

Example: loop to substract 1 65536 times — to take times

```
        mvt  r0, #0xFF00
        add  r0, #0xFF      r0 = 0xFFFF16 = 6553610
LOOP:   sub  r0, #1         r0 ← r0 - 1
        bne  LOOP           when result = 0, process to next step
```