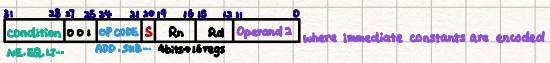
A brief note on **Immediate Mode (constants) in ARM Instructions**

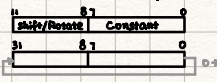
e.g: MOV R3, #6 R3 ← 6 "Immediate" address mode

- All ARM instructions are encoded into 32-bit "machine code" ("object code")



These 12 bits of operand2 don't encode constant as $0 \to 2^{12}-1$ or 2's compliment form

- Instead: 12 divide into 2 parts:



Connection To Synchronization with the outside world (i.e I/O)

The Input/Output so far:

- use input device → key 0-3, switch 9-0

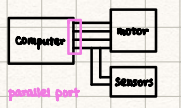
access these through special memory address - "memory mapped I/O"

- Similar for output: LEDs, HEX

- Full computer systems have many kinds of I/O devices

Polling vs. Interrupt-driven Synchronization for I/O

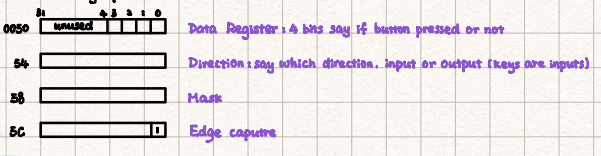
- **Polling**: processor asks device if it has new input or has finished transmitting previous output: so that processor can proceed
  - asks over and over (inefficient)
- **Interrupt-Driven**: processor stops what it is doing & runs different code to "service the interrupt"



→ Arduino / Raspberry Pi

Example: Parallel Port Interface on DE1-SoC

for the 4 key "push buttons"



Program: take a 7-bit pattern & rotates that pattern through 4 HEX display

- a new pattern can be input on switches → LEDS
- when a button is pushed & released, the new pattern is displayed. (Use polling to detect push & release of button)

code:

```
_start: LDR  R0, =0xFF200000   // base address for LEDs
        MOV  R1, 0x01          //pattern to start
Display: LDR  R4,[R0, #0x40]   // FF200040 is switches → read switches
        STR  R4, [R0]          // Display on LEDs
  //Poll to see if a key pushbutton pressed
        LDR  R5,[R0,#0x50]     // FF200050 is the key/pushbutton
        CMP  R5, #0            //if R5 is non-zero a button is pushed
        BEQ  Display
        MOV  R1, R4            // Button was pressed "capture" the new switch code into R1
  // Poll to wait for button to be released "Polling loop"
  wait: LDR  R5,[R0,#0x50]     //Read Button
        CMP  R5, #0
        BNE  WAIT
  //Button released
  NO_KEY: STR  R1,[R0,#0x20]   //Put pattern in R1 into HEX3-0
          ROR  R1, #1          //Rotate
```