Subroutines, Delay Loops, and I/O

Subroutines (aka functions, methods, procedures)

- used to perform repetitive tasks + hide details

eg: $y = f(a, b)$

- y: result
- f: subroutine name
- a, b: parameters

convention: r2, r3 for results

r4, r5, r6, r7 for parameters

important: r1 used by assembler, DO NOT USE

r31 used by CPU for return address

↳ also known as "ra" by assembler

eg: $f(a,b) = a^2 - b$

compute $f(3,4)$ and $f(f(3,4), -2)$

```
_start:  movi  r4, 3
         movi  r5, 4
         call  f          /* stores address 'here' in r31 */
here:    mov   r4, r2     /* r2 holds f(3,4) result */
         movi  r5, -2
         call  f          /* stores address 'STOP' in r31 */
STOP:    br    STOP
f:       mul   r2, r4, r4
         sub   r2, r2, r5
         ret
```

Delays

- CPUs are very fast, humans are very slow
  - CPU clock = 50 MHz
  - each instruction takes about 1 cpu clock

$$T_{instr} = \frac{1}{f} = \frac{1}{50 \times 10^6 \text{ Hz}} = 0.02 \times 10^{-6} \text{ s} = 20 \text{ ns}$$

(0.02×10^{-6} s = 0.02 μs; 10^{-9})

  - human reaction time ~ 100 ms
    - eg, to blink LED, keep it on for ~100 ms

- use subroutine to waste time

eg:

```
delay_1s:   movia   r2, 25000000
waste_time: subi    r2, r2, 1
            bne     r2, r0, waste_time
            ret
```

(2 instr)

- how slow?

loop has 2 instr (subi, bne)

$$\text{delay} = (2 + 2 \times 25000000 + 1) \times 20 \text{ ns}$$
$$\cong 50 \times 10^6 \times 20 \times 10^{-9} \text{ s}$$
$$= 1000 \times 10^{-3} \text{ s}$$
$$= 1 \text{ s}$$

Input and Output

memory-mapped I/O

ldwio input : CPU reads from special memory address

stwio output : CPU writes to " " "

special addresses ~ 0x4800 to 0x48FF

```
.equ   IOBASE,    0x4800
.equ   LEDR,      0
.equ   LEDG,      0x10
.equ   SWITCH,    0x20
.equ   KEY,       0x30
.equ   HEX7SEG,   0x40
.equ   COUNTER,   0x50
```

eg:
green LEDs are accessed at address
IOBASE + LEDG = 0x4810

examples

① turn on all ODD red LEDs (1, 3, 5, 7, 9)
by writing %10 1010 1010 = 0x2AA
to address IOBASE + LEDR = 0x4800

```
_start:  movia   r23, IOBASE
         movi    r2, 0x2AA
         stwio   r2, LEDR(r23)
```

output is done here,

LEDs will remember this pattern until changed with another stwio

① r23 can be used for all I/O
→ don't need to use one register for each I/O device address

② "LEDR" part makes it clear which device we are accessing

② continuously read input switches and copy to red LEDs

```
_start:   movia   r23, IOBASE
LOOP:     ldwio   r2, SWITCH(r23)
          stwio   r2, LEDR(r23)
          br      LOOP
```

③ if switch #3 is ON, add 1 to r2

```
_start:   movia   r23, IOBASE
LOOP:     ldwio   r3, SWITCH(r23)
          andi    r3, r3, 0x8
          beq     r3, r0, skip
          addi    r2, r2, 1
skip:     {
```

...010001
↑ ↑
#3 #0

this step is called masking