Digital Systems (Sheet #4)

Marius Gavrilescu

10.8
```
PROCEDURE divmod(a, b : INTEGER) : (INTEGER, INTEGER);
VAR d;
BEGIN
  d := 0;
  WHILE a >= b DO
    d := d + 1;
    a := a - b;
  END;
  RETURN (d, b);
END divmod;
```

```
divmod: clr $v0
loop:   blt $a0, $a1, end
        addi $v0, $v0, 1
        sub $a0, $a0, $a1
        b loop
end:    move $v1, $a0
        jr $ra
```

Efficient solution:

```
divmod: li $t0, 0                   -- q
        move $t1, $a0               -- r
        move $t2, $a1               -- y

fl:     bltu $t1, $t2, sl
        sll $t2, $t2, 1
        b fl

sl:     beq $t2, $a1, slend
        sll $t0, $t0, 1
        srl $t2, $t2, 1
        blt $t1, $t2, sl
        addi $t0, $t0, 1
        sub $t1, $t1, $t2
        b sl

slend:  move $v0, $t0
        move $v1, $t1
        jr $ra
```

This solution will loop endlessly in the first loop when the divisor is too large or we are dividing by 0. It will not give the correct result for negative numbers.

10.10
```
PROCEDURE prime(a : INTEGER);
VAR b, d, m;
BEGIN
  b := 2;
  d := a;
  WHILE d >= b DO
    (d, m) = divmod(a, b);
    IF m = 0 THEN RETURN FALSE END;
    b := b + 1;
  END
  RETURN TRUE;
END prime;
```

```
prime:  subi $sp, $sp, 4
        sw $ra, 0($sp)
        li $a1, 2
        move $v0, $a0

ploop:  blt $v0, $a1, pend
        jal divmod
        addi $a1, $a1, 1
        bne $v1, $0, ploop
        li $v0, 0
        b pret
pend:   li $v0, 1
pret:   lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra
```

12.1 `ldl n` can be implemented as follows:

```
sp
li n
add
ld
```

12.2 I see no obvious reason.

13.1 1 $\frac{1}{2}$ is 0011100000000000.
$\frac{7}{8}$ is 0011101100000000.
1 is 0011110000000000.
$\frac{3}{2}$ is 0100001000000000.
2 is 0100000000000000.
$\frac{13}{2}$ is 0100111010000000.

2 For 0.2, the closest number is $2^{-3} \cdot 1.1000110011_2$.
For 0.6, the closest number is $2^{-1} \cdot 1.0110011001_2$.

3 Smallest is 2^{-14}, largest is $2^{15} - 2^{-24}$.

4 $0.0000000001_2 \times 2^{-14}$

5 $\varepsilon = 2^{-10}$

13.3

13.6