XE36SKD

Exercise

AVR Instruction Set

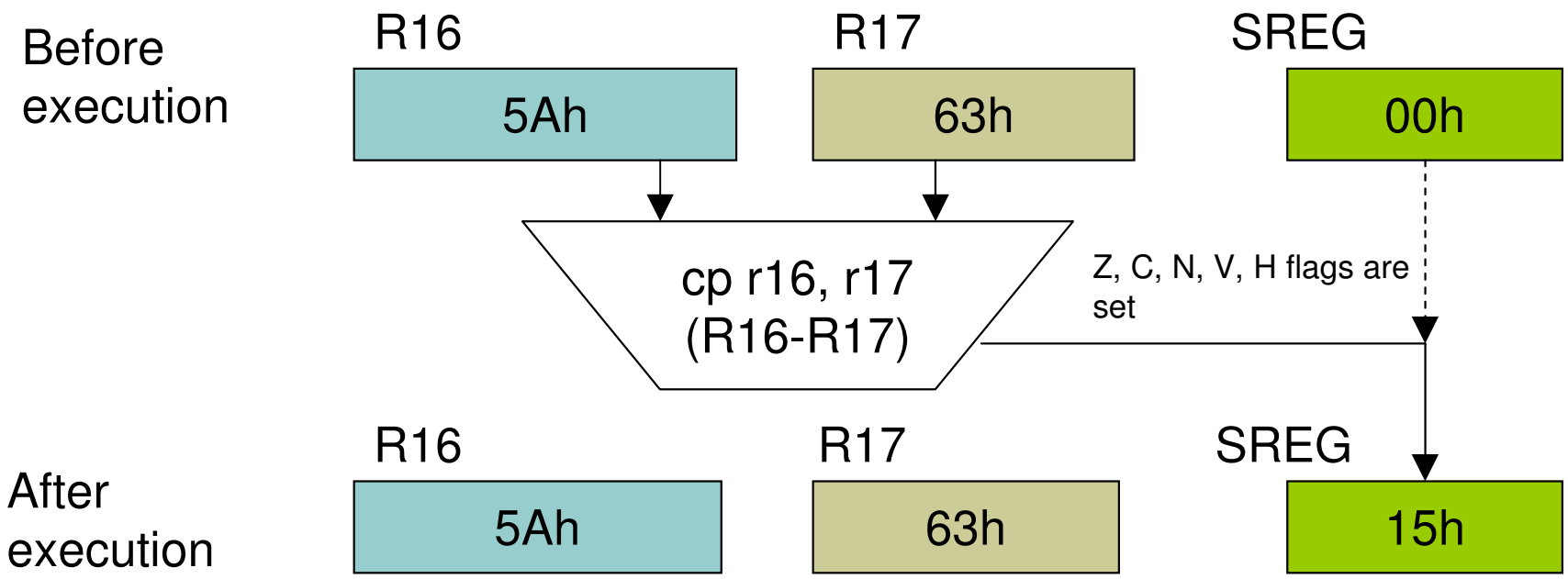
Loops, Multiplication and Division Algorithms

Miroslav Skrbek

Exercise 6

- Loops
- Multiplication Algorithm
- Division Algorithm

Comparison




```
cpi r16, 10 ; compare r16 to constant 10
cp  r1, r2  ; compare r1 to r2
cpc r3, r2  ; compare r3 to r4, include the carry bit to comparison
```

Branching the program according to the result of comparison

Branch if equal
(Z=1)

```
cp r16, r17
breq  is_equal
not_equal:
...
```

Relative address

```
breq rel ; branch if equal             Z = 1
brne rel ; branch if not equal         Z = 0
brsh rel ; branch if same or higher    C = 0   (unsigned number comparison)
brlo rel ; branch if lower             C = 1   (unsigned number comparison)
brmi rel ; branch if minus             N = 1
brpl rel ; branch if plus              N = 0
brge rel ; branch if greater or equal  S = 0   (signed number comparison)
brlt rel ; branch if less than         S = 1   (signed number comparison)
```

For loop

In Java

```
for(i=0; i < 10; i++) {
… body …
}
```

Solution I

```
        clr     r16
cycle:  cpi     r16, 10
        breq    endofcycle
        … body …
        inc     r16
        jmp     cycle
endofcycle: …
```

Solution II

```
        ldi     r16, 10
cycle:
        … body …
        dec     r16
        brne    cycle
        …
```

While loop

In Java

```
while(i < j) {
… body …
}
```

Solution I

```
R16 is i, R17 is j

cycle: cp      r16, r17
       brsh    endofcycle

       … body …
       jmp     cycle
endofcycle: …
```

Do ... while loop

In Java

```
do {
… body …
} while(i < j)
```

```
R16 is i, R17 is j

cycle:
        … body …
        cp      r16, r17
        brlo    cycle
        …
```

Empty cycle – wait loop

Solution I

```
        ldi     r16, 100
cycle:  dec     r16
        brne    r16, cycle
        …
```

Solution II

```
        clr     r16
        ldi     r17, 100
cycle:  inc     r16
        cpse    r16, r17
        jmp     cycle
        …
```

Nested loops

```
        ldi     r17, 100
c1:     clr     r16
c2:     dec     r16
        brne    c2
        dec     r17
        brne    c1
        jmp     cycle
        …
```

Task: write a program shifting a 16-bit number by n bits to the left

```
ldi     r16, 0x3A
ldi     r17, 0x53
ldi     r18, 3

… write instructions here …
```

The 16-bit number is in the R16 and R17 registers (value: 533Ah)

The number of shifts (n) is in the R18 register

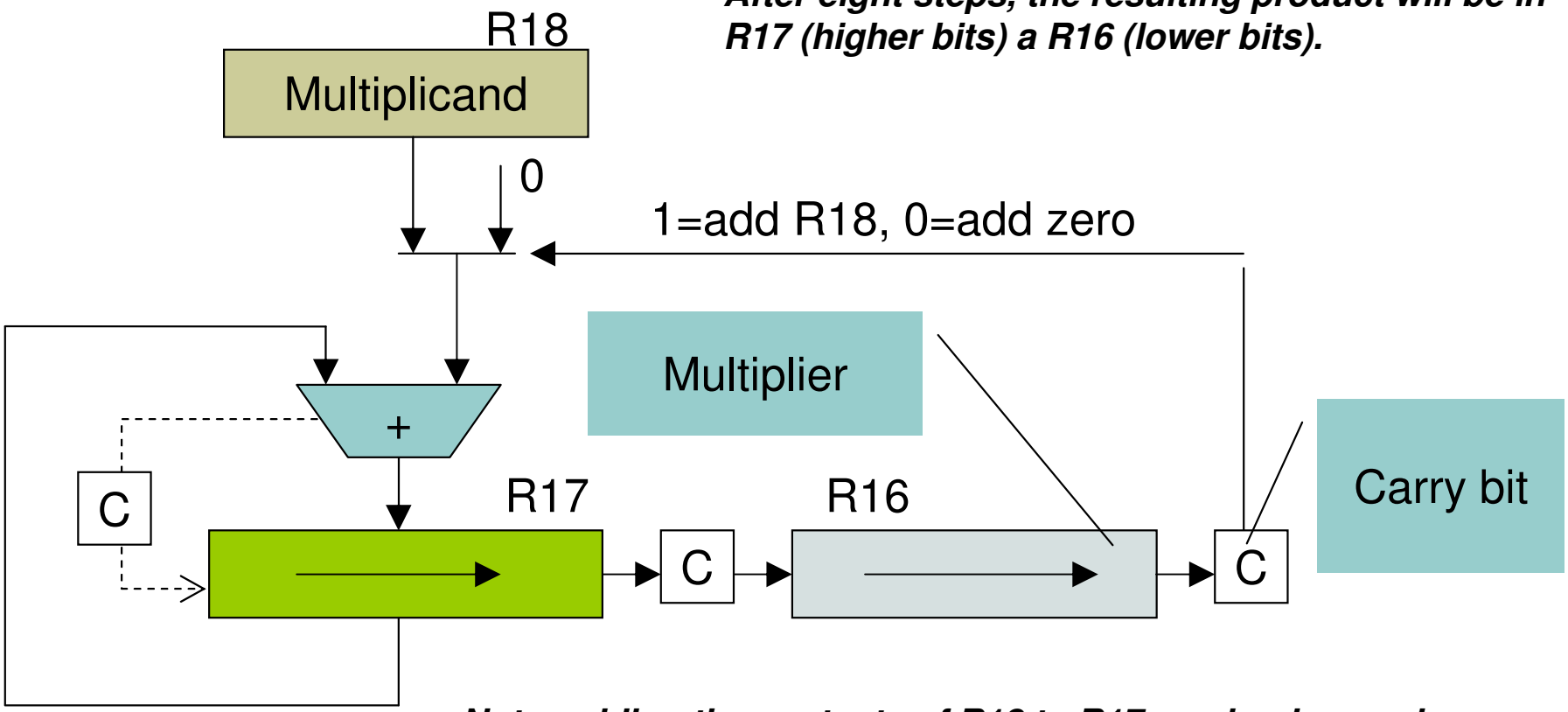
Task: multiply two 8-bit numbers

```
LDI    R16,  10  ; multiplier
LDI    R17,  100  ; multiplicand

… write instructions here …

Store the product in R16 and R17
registers:
R17 (higher bits) a R16 (lower bits)
```

13 instructions altogether

Hints

After eight steps, the resulting product will be in R17 (higher bits) a R16 (lower bits).



Note: adding the contents of R18 to R17 can be done using `add R17, R18`. Adding zero to R17 simply means skipping over the instruction `add R17, R18`.

Task: divide two eight-bit numbers

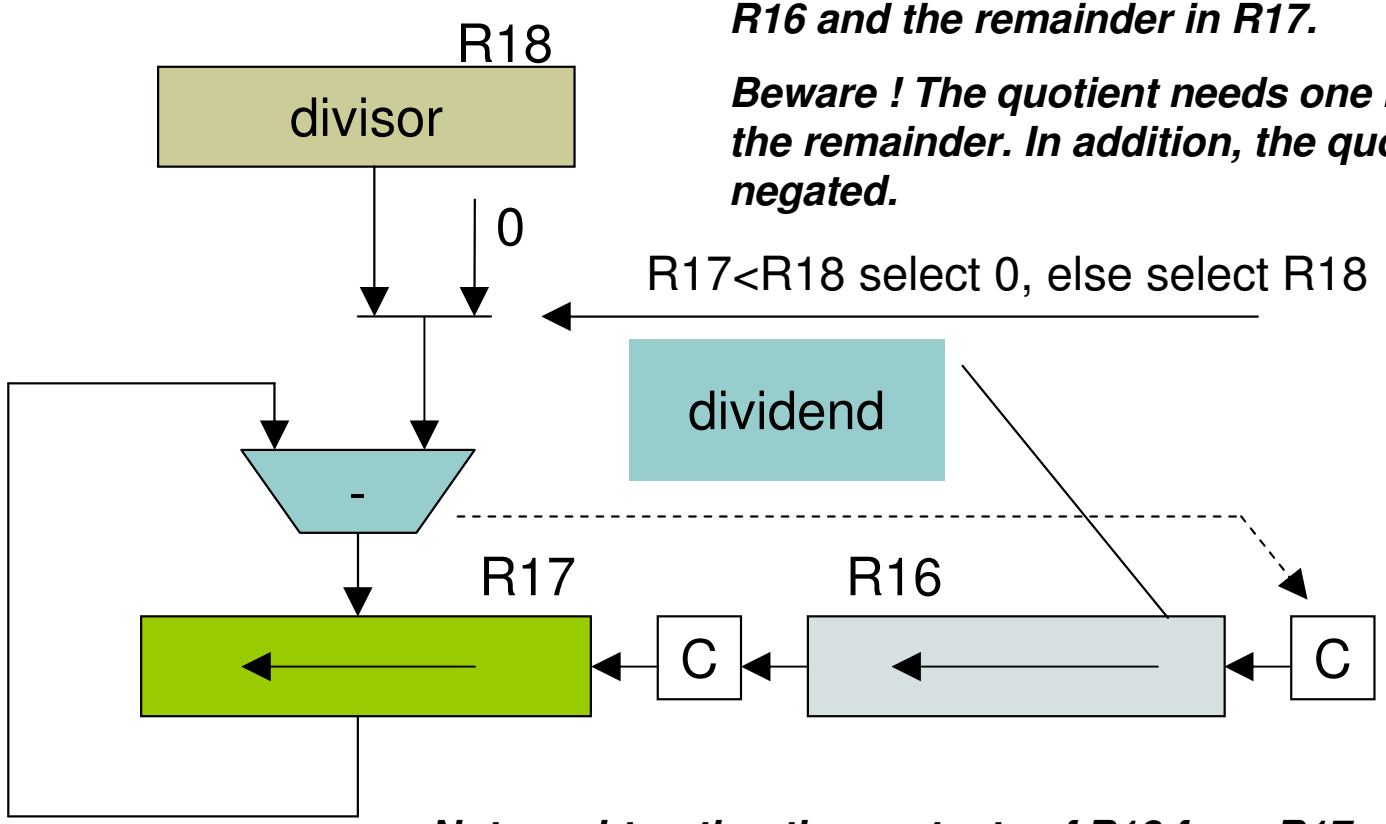
```
LDI    R16,  111   ; dividend
LDI    R17,   10   ; divisor

… write your instructions here …

Store the result:
R17 (remainder) a R16 (quotient)
```

15 instructions altogether

Hints

After eight steps, the resulting quotient will be in R16 and the remainder in R17.

Beware ! The quotient needs one more shift than the remainder. In addition, the quotient has to be negated.



Note: subtracting the contents of R18 from R17 can be done using `sub R17, R18`. Subtracting zero means simply skipping the instruction `sub R17, R18`.