6.24)   
start:  
ldi zl, low(2\*x)  
ldi zh, high(2\*x)

Clr r2

Loop:

Lpm r16, z+  
mov r18, r16  
mul r16, r16  
movw r20, r0  
mul r16, 2  
add r20, r0  
adc r21, r1  
subi r20, -5  
adc r21, r2  
cpi r16, 0  
brne loop  
  
x: .db 1,2,3,4,5,6,7,8,9,0

3.4) relative jump, one byte

3.5) jump, two byte

3.6) The instruction only takes one clock cycle to process

3.8) False

3.11) True

5.20) Branches to the position, relative to program counter, if the C-bit is 0

5.22)

a) It would branch  
 b) It would not branch

5.23) R20 is 0x85 after each program. CP instructions do not change the registers.

5.26)

start:

Ldi r16, 0x68  
clr r17  
clr r18

Loop:  
 ror r16  
 brcs has0  
 inc r18

Has0:

inc r17  
 cpi r17, 7  
 brne loop

End: rjmp end

5.27)  
start:

Ldi r16, 0x68  
clr r17

Loop:  
 ror r16  
 brcs end  
 inc r17  
 cpi r17, 7  
 brne loop

End: rjmp end

5.28)   
start:

Ldi r16, 0x68  
clr r17

Loop:  
 rol r16  
 brcs end  
 inc r17  
 cpi r17, 7  
 brne loop

End: rjmp end

Extra 1)

1. add r1, 2

Cpi r1, 1  
brlt else

Sub r1, r3

Cp r1, r2

Brlt else

Cp r3, r1  
brlt else  
cp r3, r2  
brge else

Extra 2)

Ld r16, temp1  
ld r17, temp2  
ld r18, temp3

While:

Inc r17  
 lsl r18  
 mov r18, r0  
 cp r17, r16  
 brlo while

Extra 3)

Cp r2, r1  
 brlt end

While:  
 cp r2, r3  
 brge else  
 mov r1, r2  
 rjmp endif

Else:

Mov c3, c2  
endif:

Inc r1  
 cp r2, r3  
 brlt while  
end: rjmp end