1.

#sw Rd,immed

65: addrsel=pc imload=1

addrsel=immed datawrite=1 datasel=dreg, goto pcincr

- addrsel=pc , damo kontrolni signal da pošljemo vrednost PC na podatkovno vodilo
- imload=1, v takojšni register shranimo takojšni operand
- addrsel=immed, na podatkovno vodilo damo naslov ki smo si ga korak prej zapovnili
- datawrite=1, omogočimo pisanje po pomnilniku
- datasel=dreg, na podatkovno vodilo pošljemo vrednost od dreg
- goto pcincr, števec še dodatno povečamo za ena, ker smo za ukaz rabili 2 16-bit registra

2.

```
1  # This program uses the instructions defined in the
2  # basic_microcode file. It adds the numbers from 100
3  # down to 1 and stores the result in memory location 256.
4  # (c) GPL3 Warren Toomey, 2012
5  #
6  main:   li   r1, 7      # r1 is current value
7  start:  li   r0, 13     # r0 is the loop counter
8  left:   sw   r1, 49152      # Save current value to BIN_LED
9          lsli r1,r1,1
10         subi r0,r0,1
11         jnez r0,left
12         sw   r1, 49152
13         sw   r1, 49152  #priblizno izenacim cas na obeh strajnih mestih
14         li   r0, 14
15 right:  sw   r1, 49152
16         lsri r1,r1,1
17         subi r0,r0,1
18         jnez r0,right
19         li   r1,7
20         jnez    r1, start       # loop if r1 != 0 -> loop forever
```

Zapis delovanja ukazov bo naslednji: številka vrstice + opis:

6. nalozi vrednost 7 v regidster r1 – 2 urini periodi

7. podobno kot zgoraj

8. sw, zapiše vrednost registra r2 na naslov ki pripada moji vhodnoizhodni napravi, 3 urine periode

9. lsl, število v r1 shifta v levo za 1, 3-urine periode

10. subi, od r0 odštejemo 1 oz. števec zmanjšamo za 1, 3 urine periode

11. jnez, preverimo pogoj in če zanke še nismo naredili 13-krat se vrnemo na začetek, 3 urine periode

12.,13. sw, enako kot pri 8. le da to naredimo da malce izenačimo čas , 2* 3 urine periode

14. li, steveč ponovno postavimo na 14, da bomo imeli enako število korakov v obe smeri, 2 urini periodi

15. sw, ponovno zapišemo vrednost na LED diode, 3-urine periode

16. lsri, vrednost v r1 shiftamo v levo za 1, 3-urine periode

17. subi, ponovno zmanjšamo števec, 3-urine periode

18. jnez, preverimo stevec in se vrnemo če števca še nismo spravili na 0, 3 urine periode

19. li, ponovno naložimo začetno vrednost da bomo z njo začeli, 2 urini periodi

20. jnez, vračanje nazaj na začetek programa, da Nightrider ves čas deluje, 3 urne periode

Za vsakim ukazom porabimo še 2 urini periodi za Fetch.

3. Realizirani ukazi

sub Rd,Rs,Rt

1: aluop=sub op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=-, ALU 2nd op = treg, write from aluout to reg

mul Rd,Rs,Rt (2)

2: aluop=mul op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=*, ALU 2nd op = treg, write from aluout to reg

div Rd,Rs,Rt (3)

3: aluop=div op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=/, ALU 2nd op = treg, write from aluout to reg

rem Rd,Rs,Rt (4)

4: aluop=rem op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=%, ALU 2nd op = treg, write from aluout to reg

#and Rd,Rs,Rt (5)

5: aluop=and op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=and, ALU 2nd op = treg, write from aluout to reg

#or Rd,Rs,Rt (6)

6: aluop=or op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=or, ALU 2nd op = treg, write from aluout to reg

#xor Rd,Rs,Rt (7)

7: aluop=xor op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=xor, ALU 2nd op = treg, write from aluout to reg

#nand Rd,Rs,Rt (8)

8: aluop=nand op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=nand, ALU 2nd op = treg, write from aluout to reg

#nor Rd,Rs,Rt (9)

9: aluop=nor op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=nor, ALU 2nd op = treg, write from aluout to reg

#not Rd,Rs (10)

10: aluop=not op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=not, ALU 2nd op = treg, write from aluout to reg

#lsl Rd,Rs,Rt (11)

11: aluop=lsl op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=lsl, ALU 2nd op = treg, write from aluout to reg

#lsr Rd,Rs,Rt (12)

12: aluop=lsr op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=lsr, ALU 2nd op = treg, write from aluout to reg

#asr Rd,Rs,Rt (13)

13: aluop=asr op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=asr, ALU 2nd op = treg, write from aluout to reg

#rol Rd,Rs,Rt (14)

14: aluop=rol op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=rol, ALU 2nd op = treg, write from aluout to reg

#ror Rd,Rs,Rt (15)

15: aluop=ror op2sel=treg dwrite=1 regsrc=aluout, goto fetch # ALU=ror, ALU 2nd op = treg, write from aluout to reg

#addi Rd,Rs,immed (16)

16: addrsel=pc imload=1

aluop=add op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#subi Rd,Rs,immed (17)

17: addrsel=pc imload=1

aluop=sub op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#muli Rd,Rs,immed (18)

18: addrsel=pc imload=1

aluop=mul op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#divi Rd,Rs,immed (19)

19: addrsel=pc imload=1

aluop=div op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#remi Rd,Rs,immed (20)

20: addrsel=pc imload=1

aluop=rem op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#andi Rd,Rs,immed (21)

21: addrsel=pc imload=1

aluop=and op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#ori Rd,Rs,immed (22)

22: addrsel=pc imload=1

aluop=or op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#xori Rd,Rs,immed (23)

23: addrsel=pc imload=1

aluop=xor op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#nandi Rd,Rs,immed (24)

24: addrsel=pc imload=1

aluop=nand op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#nori Rd,Rs,immed (25)

25: addrsel=pc imload=1

aluop=nor op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#lsli Rd,Rs,immed (26)

26: addrsel=pc imload=1

aluop=lsl op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#lsri Rd,Rs,immed (27)

27: addrsel=pc imload=1

aluop=lsr op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#asri Rd,Rs,immed (28)

28: addrsel=pc imload=1

aluop=asr op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#roli Rd,Rs,immed (29)

29: addrsel=pc imload=1

aluop=rol op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#rori Rd,Rs,immed (30)

30: addrsel=pc imload=1

aluop=ror op2sel=immed dwrite=1 regsrc=aluout, goto pcincr

#jeq Rs,Rt,immed (33)

33: addrsel=pc imload=1

aluop=sub op2sel=treg, if z then jump else pcincr

#jne Rs,Rt,immed (34)

34: addrsel=pc imload=1

aluop=sub op2sel=treg, if z then pcincr else jump

#jgt Rs,Rt,immed (35)

```
35:         addrsel=pc imload=1
aluop=sub  op2sel=treg, if n then pcincr else jump
#jlt Rs,Rt,immed (37)
37:         addrsel=pc imload=1
aluop=sub  op2sel=treg, if n then jump else pcincr
#jeqz Rs,immed (39)
39:         addrsel=pc imload=1
aluop=sub  op2sel=const0, if z then jump else pcincr
```

4.

Rabili bi:

-register namenjen za skladovni kazalec SP (R6)

-register namenjen za povratni naslov LR (R7)

-ukaz BL immed, za preklop na podprogram

-ukaz BX za izhod iz podprograma

-ukaza PUSH/PUL Rx, za dodjanje na sklad

BL immed, ukaz bi storil naslednje:

-trenutno vrednost PC bi si shranil v LR

-v PC bi vstavil naslov immed

BX, bi naredil naslednje:

-vrednost iz LR bi prepisal v PC

PUSH/PULL Rd, bi naredil naslednje (privzamemo full descending):

- SP-1(PUSH) / SP+1(PULL)

-SW Rd, SP   (PUSH) / RW Rd, SP (PULL)

RW Rd, Rs ukaz bi:

-sprožil podatkovni signal dataread

-na naslovno vodilo poslal naslov Rs

-na podatkovno vodilo poslal Rd

DODATNO:

-Povezani izhodni napravi

-Moja izhodna naprava z 16-bit registrom,

uporabljena s programom test_nightrider.s, v katerem so uporabljeni tudi drugi ukazi