2.5.12. Dijeljenje uzastopnim oduzimanjem i rastavljanje na proste faktore

Rješenje:

```
        ;traže se samo potprogrami, negdje prije je inicijaliiran stog
DIV
                PUSH  R2            ;spremanje konteksta
                MOVE  -1,  R2
LOOP                                ;dijeljenje
                ADD R2, 1, R2
                SUB R0, R1, R0
                JR_NN LOOP

                ADD R0, R1, R1      ;R1 = ostatak
                MOVE R2, R0         ;R0 = rezultat

                POP R2              ;obnova konteksta
                RET

FACT
                PUSH R0             ;spremanje konteksta
                PUSH R1
                PUSH R2
                PUSH R3
                PUSH R4
                MOVE 2, R2          ;prvi prosti faktor

PETLJ
                MOVE R0, R3         ;spremanje broja
                MOVE R1, R4         ;spremanje adrese
                MOVE R2, R1         ;R1 djelitelj
                CALL DIV
                CMP R0, 0           ;ako je rezultat 0, djelitelji su preveliki da bi bili faktori
                JR_EQ VAN
                CMP R1, 0           ;ako nema ostatka onda je broj djeljiv
                JR_EQ DIJELI

NEDIJELI
                MOVE R3, R0         ;ako ima, vracamo vrijednost broju
                MOVE R4, R1         ;vracamo adresu
                ADD R2, 1, R2       ;povecavamo djelitelj
                JR PETLJ
DIJELI
                MOVE R4, R1         ;ako djeli, vracamo dresu
                STORE R2, (R1)      ;spremamo faktor
                ADD R1, 4, R1       ;povecavamo adresu
                JR PETLJ
VAN
                POP R4              ;obnova konteksta
                POP R3
                POP R2
                POP R1
```

```
                POP R0
                RET
```

2.5.13 Oduzimanje i dijeljenje brojeva u dvostrukoj preciznosti

Rješenje:

```
                `ORG 0
MAIN
                MOVE 10000, SP          ;inicijalizacija stoga
                MOVE PRVI, R0           ;adresa djeljenika
                MOVE DRUGI, R1          ;adresa djelitelja
                MOVE REZ, R2            ;adresa rezultata
                CALL DIV_DBL
                HALT

PRVI            DW ...                  ;neki broj u dvostrukoj preciznosti
DRUGI           DW ...                  ;neki broj u dvostrukoj preciznosti
REZ             `DS 8

DIV_DBL
                PUSH R3                 ;spremanje konteksta
                PUSH R4
                MOVE -1, R3
LOOP                                    ;dijeljenje...
                ADD R3, 1, R3
                CALL SUB_DBL
                LOAD R4, (R2 + 4)       ;...ako je rezultat negativan, onda je kraj
                ROTL R4, 1, R4          ;predznak u najvisem bitu (63. bit)
                AND R4, 1, R4
                JR_Z LOOP

                MOVE R3, (R2)           ;rezultat je u jednostrukoj preciznosti, što je mozda
                POP R4                  ;greška ??
                POP R3
                RET

SUB_DBL
                PUSH R3                 ;spremanje konteksta
                PUSH R4
                PUSH R5
                PUSH R6
                LOAD R3, (R0)           ;dohvat operanada
                LOAD R4, (R0 + 4)
                LOAD R5, (R1)
                LOAD R6,(R1 + 4)

                SUB R3, R5, R3          ;oduzimanje, valjda tocno
                SBC R4, R6, R4

                STORE R3, (R2)          ;spremanje rezultata
                STORE R4, (R2 + 4)
```

```
        POP R6              ;obnova konteksta
        POP R5
        POP R4
        POP R3
        RET
```

2.5.14. Potprogram za izračun funkcije sa četiri parametra

Rješenje:

```
        `ORG 0
MAIN
        MOVE 10000, SP      ;inicijalizacija stoga
        MOVE 1000, R6       ;adresa bloka
LOOP
        LOAD R0, (R6)       ;ucitavanje podataka
        CALL FUNC
        STORE R0, (R6)      ;spremanje rezultata
        ADD R6, 4, R6       ;povecavanje adrese, 4 jer spremamo 32 bita
        CMP R6, 2000        ;je li kraj
        JR_NE LOOP          ;ako nije nastavi
        HALT
FUNC
        PUSH R1             ;spremanje konteksta
        PUSH R2
        PUSH R3
        MOVE 0, R2          ;R2 = y
        MOVE 0, R3          ;R3 = broj pomaka, tj. množenje
PETLJ
        CMP R3, 4           ;ako je kraj onda van
        JR_EQ VAN
        AND R0, 0FF, R1     ;najnizi bajt
        SHL R1, %D 24, R1   ;predznacno prosirivanje
        ASH R1, %D 24, R1
        SHL R1, R3, R1      ;mnozenje s potencijom od 2, prvo 2^0, pa na 2^1 itd.
        ADD R1, R2, R2      ;povecavanje rezultata
        ADD R4, 1, R4       ;povecavanje pomaka
        JR PETLJ
VAN
        MOVE R2, R0         ;rezultat se vraca preko R0
        POP R3              ;obnova konteksta
        POP R2
        POP R1
        RET
```

2.5.15. Zrcalni bitovi i provjera palindroma

Rješenje:

```
        `ORG 0
MAIN
```

```
        MOVE 10000, SP
        MOVE 500, R0            ;stavljanje adrese na stog
        PUSH R0
        MOVE %D 512, R0         ;stavljanje broja podataka na stog
        PUSH R0
        CALL PALIN
        ADD SP, 8, SP           ;brisanje argumenata sa stoga
        STORE R0, (450)         ;spremnje rezultata
        HALT

PALIN
        PUSH R1                 ;spremanje konteksta
        PUSH R2
        PUSH R3
        PUSH R4
        MOVE 0, R4              ;R4 brojac palindroma
        LOAD R2, (SP + %D 20)   ;R2 broj podataka
        LOAD R1, (SP + %D 24)   ;R1 adresa podataka
LOOP
        LOAD R0, (R1)           ;ucitavanje podata
        MOVE R0, R3             ;kopiranje podatka
        CALL INVERZ
        CMP R0, R3              ;ako je inverz jednak broju onda je palindrom
        JR_NE NISU
JESU
        ADD R4, 1, R4           ;ako je palindrom povecaj brojac
NISU
        ADD R1, 4, R1           ;povecavanje adrese
        SUB R2, 1, R2           ;smanjenje broja podataka
        JR_NZ LOOP              ;ako ih ima nastavi

        MOVE R4, R0             ;rezultat se vraca u R0
        POP R4                  ;obnova konteksta
        POP R3
        POP R2
        POP R1
        RET

INVERZ
        PUSH R1                 ;spremanje konteksta
        PUSH R2
        PUSH R3
        MOVE %D 32, R3          ;brojac za petlju
        MOVE 0, R2              ;brisanje registra rezultata
PETLJ
        AND R0, 1, R1           ;najnizi bit broja stavimo u R1
        SHR R0, 1, R0           ;pomaknemo broj u desno
        OR R1, R2, R2           ;stavimo taj bit u R2
        SHL R2, 1, R2           ;i pomaknemo R2 u lijevo
        SUB R3, 1, R3           ;i tako 32 puta
        JR_NZ PETLJ
```

```
                MOVE R2, R0             ;rezultat je u R0
                POP R3                  ;obnova konteksta
                POP R2
                POP R1
                RET
```

2.5.16. Potprogram za pretvorbu iz zapisa BCD u

Rješenje:

```
                `ORG 0
MAIN
                MOVE 10000, SP
                MOVE 2000, R6           ;adresa
XYZ
                LOAD R0, (R6)           ;ucitavanje podatka
                CALL BCD2NBC
                STORE R0, (R6)          ;spremanje rezultata
                ADD R6, 4, R6
                CMP R6, 2500            ;ako je kraj,van
                JR_NE XYZ
                HALT

BCD2NBC
                PUSH R1                 ;spremanje konteksta
                PUSH R2
                PUSH R3
                MOVE 4, R3              ;R3 brojac za petlju
                MOVE 0, R2              ;R2 rezultat
LOOP
                ROTL R0, 8, R0          ;najvisi bajt na donjih 8 bitova
                AND R0, 0FF, R1         ;izdvojimo ga
                ADD R1, R2, R2          ; dodamo ga u R2
                SHL R2, 3, R1           ;pomnozimo s 8
                ADD R1, R2, R1          ;dodamo i...
                ADD R1, R2, R2          ;dodamo,dakle *10
                SUB R3, 1, R3           ;ponavljamo za sve znamenke
                JR_NZ LOOP

                MOVE R2, R0             ;rezultat u R0
                POP R3                  ;obnova konteksta
                POP R2
                POP R1
                RET
```

2.5.17. Brojenje pojavljivanja 2-bitnog uzorka u broju

greška je u tekstu zadatka, ne pomice se za 2 nego za 1 sto je i ocito iz primjera

Rješenje:

```
                `ORG 0
```

```
MAIN
        MOVE 10000,SP       ;inicijalizacija stoga
        MOVE 1000, R3       ;R3 adresa izvora
        MOVE 1500, R4       ;R4 adresa odredišta

XYZ
        LOAD R0, (R3)       ;ucitavanje podatka

        MOVE %B 00, R1      ;prvi uzorak
        CALL CNT2BIT
        STOREB R2, (R4)

        MOVE %B 01, R1      ;drugi uzorak
        CALL CNT2BIT
        STOREB R2, (R4 + 1)

        MOVE %B 10, R1      ;treci uzorak
        CALL CNT2BIT
        STOREB R2, (R4 + 2)

        MOVE %B 11, R1      ;cetvrti uzorak
        CALL CNT2BIT
        STOREB R2, (R4 + 3)

        ADD R4, 4, R4       ;povecanje adresa
        ADD R3, 4, R3
        CMP R3, 1200        ;uspoedba s krajem
        JR_NE XYZ
        HALT

CNT2BIT
        PUSH R0             ;spremanje konteksta
        PUSH R3
        PUSH R5
        MOVE %D 31, R3      ;broj pomaka
        MOVE 0, R2          ;broj pojavljivanja
LOOP
        AND R0, %B 11, R5   ;izdvajanje 2 najniza bita
        CMP R5, R1          ;jesu li jednaki uzorku
        JR_NE NISU
JESU                        ;ako jesu povecati brojac
        ADD R2, 1, R2
NISU
        SHR R0, 1, R0       ;pomicati broj za JEDAN u desno
        SUB R3, 1,R3        ;i tako 31 put
        JR_NZ LOOP

        POP R5              ;obnova konteksta
        POP R3
        POP R0
        RET
```

2.5.18. Pronalazak podniza u nizu bitova

Rješenje:

```
        `ORG 0
MAIN
        MOVE 10000, SP      ;inicijlizacija stoga
        MOVE 500, R6        ;adresa bloka
        MOVE 0A1, R1        ;uzorak
XYZ
        LOAD R0, (R6)       ;ucitavanje podataka
        CALL SUBS
        CMP R0, 0           ;ako nema, oba djela su 0 pa je sve 0
        JR_NE IMA
NEMA
        MOVE 0, R0          ;ako nema briše se
        STORE R0, (R6)
IMA
        ADD R6, 4, R6       ;povecanje adrese
        CMP R6, 1000        ;ako kraj van
        JR_NE XYZ
        HALT

SUBS
        PUSH R2             ;spremanje konteksta
        PUSH R3
        PUSH R4
        PUSH R5
        MOVE %D 24, R3      ;broj pomaka
        MOVE 1, R2          ;jedinica koja ce se upisat na mjesto gdje i ako se
        MOVE 0, R4          ;pojavi uzorak, R4 rezultat
LOOP
        AND R0, 0FF, R5     ;izdvojimo najnizi bajt
        CMP R5, R1          ;provjerimo je li jednak uzorku
        JR_NE NISU
JESU                        ;ako je...
        OR R2, R4, R4       ;stavimo jedinicu na mjesto pojavljivanja
        ROTL R4, %D 26, R4  ;povecamo broj pojavljivanja
        ADD R4, 1, R4       ;pomak za 26 jer tamo pocinje brojac pojavljivanja
        ROTR R4, %D 26, R4
NISU
        SHR R0, 1, R0       ;pomicemo podataka u desno
        SHL R2, 1, R2       ;jedinicu koja pokazuje na mjesto pojavljivanja u lijevo
        SUB R3, 1, R3       ;smanjenje brojaca
        JR_NZ LOOP

        MOVE R4, R0         ;rezultat u R0
        POP R5              ;obnova konteksta
        P0P R4
        POP R3
        POP R2
        RET
```

2.5.19. Zbrajanje brojeva u zapisu nepomičnog zareza

Rješenje:

```
            `ORG 0
MAIN
            MOVE 10000, SP          ;inicijalizacija stoga

            LOADB R1, (600)         ;prvi operand
            PUSH R1
            LOADB R1, (601)         ;drugi operand
            PUSH R1
            CALL ZBR_PLD            ;zbroji, rezultat u R0
            ADD SP, 8, SP           ;brisi argumente

            LOADB R1, (602)         ;treci operand
            PUSH R0                 ;zbroj prva dva
            PUSH R1                 ;treci
            CALL ZBR_PLD            ;zbroji, rezultat u R0
            ADD SP, 8, SP           ;brisi argumente
            STORE R0, (500)         ;spremi rezultat
            HALT

ZBR_PLD
            PUSH R1                 ;spremanje konteksta
            PUSH R2
            PUSH R3
            PUSH R4

            LOAD R1, (SP + %D 20)   ;R1 argument
            LOAD R3, (SP + %D 24)   ;R3 argument
            AND R1, 0F, R2          ;decimalni dio prvog u R2
            AND R3, 0F, R4          ;decimalni diodrugog u R4
            SHR R1, 4, R1           ;dovedi cjeloborojne djelove na najniža 4 bita
            SHR R3, 4, R3           ;moze se i drugacije

            ADD R2, R4, R2          ;zbroji decimalni dio
            AND R2, 10, R4          ;ako je doslo do preljeva dodati 1 u cjelobrojni
            JR_Z NEMA_C
IMA_C
            ADD R1, 1, R1           ;dodaj 1
            AND R2, 0F, R2          ;pocisti decimalni dio
NEMA_C
            ADD R1, R3, R1          ;zbroji nedecimalni dio
            SHL R1, 4, R1           ;pomakni na njegovo mjesto
            OR R1, R2, R0           ;spoji dva dijela

            POP R4                  ;obnova konteksta
            POP R3
            POP R2
            POP R1
            RET
```

2.5.20. Izračun cjelobrojnog dijelabroja u zapisu IEEE

moguce da je ovo krivo :)

Rješenje:

```
IEEE_INT
            PUSH R0
            PUSH R2
            PUSH R3
            PUSH R4

            ROTL R0, 1, R2
            AND R2, 1, R2           ;R2 predznak

            SHR R0, %D23, R3
            AND R3, 0FF, R3

            SHL R0, 9, R0           ;R0 decimalni dio
            SUB R3, %D 127, R3      ;R3 eksponent
            PUSH R3                 ;spremanje eksponenta za kasnije, moze i u drugi reg

            CMP R3, 0
            JR_NN  NIJEM
MINUS
            MOVE 0, R1              ;ako je eksponent manji od 0, cjelobrojni dio je 0
            JR VAN
NIJEM
            JR_NE NENULA
NULA
            MOVE 1, R1              ;ako je eksponent 0, cjelobrojni dio je 1
            JR VAN
NENULA
            MOVE 1, R4
            ROTR R4, 1, R4
            MOVE  R4, R1

LOOP
            ROTL R1, 1, R1
            ASHR R4, 1, R4
            SUB R3, 1, R3
            JR_NZ LOOP

            AND R0, R4, R0          ;R0 dio decimala koji postaje cjelobrojan
            POP R3                  ;R3 eksponent
            MOVE %D 32, R4
            SUB R4, R3, R4
            SHL R0, R4, R0          ;R0 cjelorojni diO pomaknut gdje mu je mjesto
            ORR R0, R1, R1
PREDZNAK
            CMP R2, 0
            JR_EQ PLUS
MINUS
            XOR R1, -1, R1
```

```
            ADD R1, 1, R1
PLUS
            POP R4
            POP R3
            POP R2
            POP R0
            RET
```

3.3.5. Nadziranje motora pomoću CT-a

Rješenje:

```
VJP         `EQU 0FFFFFF00
VJS         `EQU 0FFFFFF04
CTLR        `EQU 0FFFFFF20
CTCR        `EQU 0FFFFFF24
CTIACK      `EQU 0FFFFFF28
CTIEND      `EQU 0FFFFFF2C
BVJ         `EQU 0FFFFFF40

            `ORG 0
            MOVE 10000, SP
            JR GLAVNI
            `ORG 8
            DW PREK
GLAVNI
            MOVE %D 50000, R0       ;konstanta za 1 s
            STORE R0, (CTLR)
            MOVE %B 10100000, SR    ;GIE =1, EINT1 = 1
            MOVE %B 11, R0          ;brojilo broji i postavlja prekide
            STORE R0, (CTCR)
LOOP
            LOAD R0, (VJS)          ;cekanje na spremnost
            OR R0, R0, R0
            JR_Z LOOP

            LOAD R0, (BROJ)         ;povecanje broja okretaja
            ADD R0, 1, R0
            STORE R0, (BROJ)
            STORE R0, (VJS)         ;dojava kraja posluzivanja

            LOAD R0, (GASI)         ;provjera je li ugasen motor
            CMP R0, 1
            JR_EQ LOOP
            HALT

BROJ        DW 0
GASI        DW 1

PREK
            PUSH R0
            MOVE SR, R0
```

```
                PUSH R0
                STORE R0, (CTIACK)          ;dojava prihvata prekida
                LOAD R0, (BROJ)
                CMP R0, %D 76
                MOVE 0, R0                  ;ako je vise od 76 gasi motor
                JR_ULE VAN
VISE

                STORE R0, (BVJ)
                STORE R0, (GASI)
                STORE R0, (CTCR)
VAN
                STORE  R0, (BROJ)           ;resetiranje broja okretaja
                STORE R0, (CTIEND)          ;dojava kraja posluzivanja
                POP R0
                MOVE R0, SR
                POP R0
                RETI
```

3.4.7. Dva sklopa PIO i uvjetna jedinica

greska je u tekstu zadatka, ne moze 32 bitni broj bit procitans 8 bitnog PIO sklopa

Rješenje:

```
VJP             `EQU 0FFFF0000
VJS             `EQU 0FFFF0004
P1UR            `EQU 0FFFF1000
P1DR            `EQU 0FFFF1004
P1IACK          `EQU 0FFFF1008
P1IEND          `EQU 0FFFF100C
P2UR            `EQU 0FFFF2000
P2DR            `EQU 0FFFF2004

                `ORG 0
                MOVE 10000, SP
                JR GLA
                `ORG 8
                DW PREK

GLA
                MOVE %B 10010000, SR        ;omoguci prekid
INITPIO
                MOVE %B011, R0              ;ulazni nacin, prekid
                STORE R0, (P1UR)
                MOVE %B 100, R0             ;postavljanje bitova, bez prekida
                STORE R0, (P2UR)

LOOP
                LOAD R0, (VJS)              ;cekanje spremnosti
                OR R0, R0, R0
                JR_Z LOOP

                LOAD R0, (BROJ)
```

```
                STORE R0, (VJP)         ;slanje broja negativnih podataka
                STORE R0, (VJS)         ;dojava kraja posluzivanja
                LOAD R0, (KRAJ)         ;provjera kraja
                CMP R0, 0
                JR_EQ LOOP
                HALT

BROJ            DW 0
KRAJ            DW 0

PREK
                PUSH R0                 ;spremanje konteksta
                MOVE SR, R0
                PUSH R0

                STORE R0, (P1IACK)      ;dojava prihvata prekida
                LOAD R0, (P1DR)         ;ucitaj podatak
                CMPR R0, 80             ;usporedi s  %B10000000
                JR_NE DALJE
JE
                MOVE  1, R0             ;ako je to je kraj
                STORE R0, (KRAJ)
VAN
                STORE R0, (P1IEND)
                POP R0
                MOVE R0, SR
                POP R0
                RETI
DALJE
                STORE R0, (P2DR)        ;ako nije šalji na PIO2
                CMP R0, 0
                JR_NN VAN
MINUS                                   ;ako je negativan povecaj brojac
                LOAD R0, (BROJ)
                ADD R0, 1, R0
                STORE R0, (BROJ)
                JR VAN
```

3.4.8. Dva sklopa PIO i bezuvjetna jedinica

Rješenje:

```
BVJ             `EQU 0FFFF0000
P1UR            `EQU 0FFFF1000
P1DR            `EQU 0FFFF1004
P1SB            `EQU 0FFFF1008
P1END           `EQU 0FFFF100C
P2DR            `EQU 0FFFF2000
P2DR            `EQU 0FFFF2004
P2SB            `EQU 0FFFF2008
P2END           `EQU 0FFFF200C
```

```
            `ORG 0
            MOVE 1000, R6

            MOVE %B 001, R0         ;PIO1 ulazni nacin bez prekida
            STORE R0, (P1UR)

            MOVE 0, R0              ;PIO2 izlazni nacin bez preda
            STORE R0, (P2UR)
LOOP1
            LOAD R0, (P1UR)         ;cekanje spremnosti PIO1
            OR R0, R0, R0
            JR_Z LOOP1

            STORE R0, (P1SB)        ;...
            LOAD R1, (P1DR)
            STORE R0, (P1END)
LOOP2
            LOAD R0, (P2UR)         ;....
            OR R0, R0, R0
            JR_Z LOOP2

            STORE R0, (P2SB)
            STORE R1, (P2DR)
            STORE R0, (P2END)

            AND R1, 1, R0
            JR_Z PARAN
NEPAR
            STORE R1, (BVJ)         ;ako je neparan posalji na BVJ
            JR KRAJ
PARAN
            STORE R0, (BVJ)         ;ako je paran posalji 0
KRAJ
            SUB R6, 1, R6           ;dok ne provjeris svih 1000 podataka
            JR_NZ LOOP1
            HALT
```

3.4.9. Sklop PIO, prekidna i uvjetna vanjska jedinica

Rješenje:

```
PUR         `EQU 0FFFF0000
PDR         `EQU 0FFFF0004
PIACK       `EQU 0FFFF0008
PIEND       `EQU 0FFFF000C
PVJPOD      `EQU 0FFFF1000
PVJIACK     `EQU 0FFFF1004
PVJIEND     `EQU 0FFFF1008
UVJDR       `EQU 0FFFF2000
UVJSR       `EQU 0FFFF2004
```

```
            `ORG 0
            MOVE 10000, SP
            JR GLAVNI
            `ORG 8
            DW PREKPIO
            `ORG 0C
            JR PREKPVJ

GLAVNI
            MOVE %B 10010000, SR        ;omoguci prekid na INT0
            MOVE %B 011, R0
            STORE R0, (PUR)
LOOP
            LOAD R0, (UVJSR)            ;cekaj spremnost
            CMP R0, 0
            JR_Z LOOP

            LOAD R0, (BROJ)             ;posalji broj podataka
            STORE R0, (UVJDR)
            STORE R0, (UVJSR)
            JR_LOOP

BROJ        DW 0
POM         DW 0

PREKPIO
            PUSH R0
            PUSH R1
            MOVE SR, R0
            PUSH R0

            STORE R0, (PIACK)
            LOAD R0, (PDR)              ;ucitaj podatak
            LOAD R1, (POM)              ;ucitaj pomak od adrese 1000
            STORE R0, (R1 + 1000)       ;spremi podatak

            ADD R1, 4, R1               ;uvecaj pomak, ako je 1000 stavi ga na 0
            CMP R1, 1000
            JR_NE NIJE
JE
            MOVE 0, R1
NIJE
            STORE R1, (POM)             ;spremi pomak

            LOAD R1, (BROJ)             ;uvecaj brojpodataka i spremi ga
            ADD R1, 1, R1
            STORE R1, (BROJ)

            STORE R1, (PIEND)           ;zavrsetak obrade prekida
            POP R0
            MOVE R0, SR
            POP R0
```

```
                POP R1
                RETI

PREKPVJ
                PUSH R0
                PUSH R1
                MOVE SR, R0
                PUSH R0

                STORE R0, (PVJIACK)             ;prihvat prekida

                LOAD R0, (POM)                  ;ucitaj pomak
                MOVE 1000, R1
                SUB R1,R0, R0                   ;ukupno mjesta je 1000 - pomak

                SHR R0, 2, R0                   ;stane mjesta/4 podataka

                STORE R0, (PVJDR)
                STORE R0, (PVJIEND)

                POP R0
                MOVE R0, SR
                POP R1
                POP R0
                RETN
```

3.5.3. DMA-prijenos iz memorije u vanjsku jedinicu (krađa ciklusa, prekid)

Rješenje:

```
DMASRC      `EQU 0FFFFFF00
DMADST      `EQU 0FFFFFF04
DMACNT      `EQU 0FFFFFF08
DMAUR       `EQU 0FFFFFF0C
DMASTART    `EQU 0FFFFFF10
DMAIACK     `EQU 0FFFFFF14
BVJ         `EQU 0FFFFFFD0
BVJSTOP     `EQU 0FFFFFFD4

                `ORG 0
                MOVE 10000, SP
                JR GLAVNI
                `ORG 8
                DW PREK

GLAVNI
                MOVE 500, R2                    ;R2 adresa bloka
                STORE R2, (DMASRC)

                MOVE BVJ, R0                    ;BVJ odrediste
                STORE R0, (DMADST)
```

```
                MOVE %D 1000, R1            ;R1 kolicina podataka
                STORE R1, (DMACNT)

                MOVE %B 1011, R0            ;dst VJ, src MEM, krađa cik, prekid
                STORE R0, (DMAUR)

                MOVE 0, R3                  ;R3 brojac negativnih
                MOVE %B 10010000, SR        ;omoguci prekid na INT0
                STORE R0, (DMASTART)

LOOP
                LOAD R0, (R2)               ;citanje podataka
                CMP R0, 0
                JR_NN PLUS
MIN
                ADD R3, 1, R3               ;ako je negativan povecaj brojac
PLUS
                ADD R2, 4, R2               ;poakni adresu
                SUB R1, 1, R1               ;smanji brojac podataka
                JR_NZ LOOP

                LOAD R1, (STANJE)           ;cekaj kraj DMA
                CMP R1, 0
                JR_NZ LOOP

                STORE R1, (BVJSTOP)         ;zaustavi BVJ
                MOVE 0, SR
                STORE R3, (400)             ;spremi rultat
                HALT

STANJE          DW 1

PREK
                PUSH R0
                STORE R0, (DMAIACK)         ;potvrda prihvata prekida
                MOVE 0, R0
                STORE R0, (STANJE)          ;obiljezi da je DMA prenio
                POP R0
                RETI
```

3.5.6. Sklop DMA i dvije bezuvjetne vanjske jedinice (krađa ciklusa, prekid)

Rješenje:

```
DMASRC      `EQU 0FFFFFF00
DMADST      `EQU 0FFFFFF04
DMACNT      `EQU 0FFFFFF08
DMAUR       `EQU 0FFFFFF0C
DMASTART    `EQU 0FFFFFF10
```

```
DMAIACK   `EQU 0FFFFFF14
BVJ1      `EQU 0FFFF0000
BVJ2      `EQU 0FFFF0004

          `ORG 0
          MOVE 10000, SP
          JR GLAVNI
          `ORG 8
          DW PREK

GLAVNI
          MOVE 1000, R1              ;R1 adresa
          STORE R1, (DMADST)

          MOVE %D 500, R2            ;R2 brojac
          STORE R2, (DMACNT)

          MOVE BVJ1, R0              ;upis izvora
          STORE R0, (DMASRC)

          MOVE %B 0111, R0           ;dst MEM, src VJ, krađa cik, prekid
          STORE R0, (DMAUR)

          MOVE %B 10010000, SR       ;omoguci prkeid na INT0
          STORE R0, (DMASTART)
LOOP
          LOAD R0, (KRAJ)            ;cekaj kraj DMA prijenosa
          OR R0, R0, R0
          JR_Z LOOP

          MOVE 0, R3                 ;brojac parnih
PET
          LOAD R0, (R1)              ;ucitaj podatak
          ADD R1, 4, R1              ;povecaj adresu
          AND R0, 1, R0
          JR_NZ NEPAR
PAR
          ADD R3, 1, R3              ;povecaj brojac ako je paran
NEPAR
          SUB R2, 1, R2              ;smanji brojac podataka
          JR_NZ PET

          STORE R3, (BVJ2)           ;spremi broj parnih
          HALT

KRAJ      DW 0

PREK
          PUSH R0
          STORE R0, (DMAIACK)        ;prihvat prekida
          MOVE 1, R0
          STORE R0, (KRAJ)           ;stavi oznaku kraja
```

```
                POP R0
                RETI
```

3.6.1. Jedna prekidna i dvije bezuvjetne vanjske jedinice

Rješenje:

```
PVJDR       `EQU  0FFFFFF20
PVJIACK     `EQU  0FFFFFF24
PVJIEND     `EQU  0FFFFFF28
BVJ1        `EQU  0FFFFFF30
BVJ2        `EQU  0FFFFFF40

                `ORG 0
                MOVE 10000, SR
                JR MAIN
                `ORG 0C
FIQ
                PUSH R0
                PUSH R2
                MOVE SR, R0
                PUSH R0

                STORE R0, (PVJIACK)     ;potvrda prihvata prekida
                LOAD R0, (PVJDR)        ;citanje podataka
                CMP R0, -1              ;provjera kraja
                JR_EQ GOTOVO
NIJR
                AND R0, 1, R2           ;provjera parnosti
                JR_NZ NEPAR
PAR
                LOAD R2, (BROJ)         ;ako paran povecati broj parnih
                ADD R2, 1, R2
                STORE R2, (BROJ)
NEPAR
                XOR R0, -1, R0          ;inace 1'k u BVJ2
                STORE R0, (BVJ2)

                ADD R0, 1, R0           ;2'k u BVJ1
                STORE R0, (BVJ1)
VAN
                POP R0
                MOVE R0, SR
                POP R2
                POP R0
                RETN
GOTOVO
                MOVE 0, R0              ;postavi oznaku kraja
                STORE R0, (KRAJ)
                JR VAN
```

```
BROJ    DW 0
KRAJ    DW 1

MAIN
        LOAD R0, (KRAJ)         ;cekaj kraj
        OR R0, R0, R0
        JR_NZ MAIN

        LOAD R0, (BROJ)
        STORE R0, (600)         ;spremi broj parnih na (600)
        HALT
```

3.6.2. Dvije prekidne i jedna bezuvjetna banjska jedinica

Rješenje:

```
PVJ1DR      `EQU 0FFFFFF20
PVJ1IACK    `EQU 0FFFFFF24
PVJ1IEDN    `EQU 0FFFFFF28
PVJ2DR      `EQU 0FFFFFF30
PVJ2IACK    `EQU 0FFFFFF34
PVJ2IEND    `EQU 0FFFFFF38
BVJ         `EQU 0FFFFFF40

            `ORG 0
            MOVE 10000, SP
            JR MAIN
            `ORG 8
            DW IRQ
MAIN
            MOVE %B 11100000, SR    ;omoguci prekide INT2, INT1
LOOP
            JR LOOP

BRZ         DW 0                    ;broj nula

IRQ
            PUSH R0
            PUSH R1
            MOVE SR, R0
            PUSH R0

            LOAD R0, (PVJ2IACK)     ;je li 2 postavio prekid
            OR R0, R0, R0
            JR_NZ OBR2              ;ako je obradi 2
OBR1                                ;inace obradi 1
            STORE R0, (PVJ1IACK)    ;prihvat prekida
            LOAD R0, (PVJ1DR)
```

```
                OR R0, R0, R0
                JR_NZ NIJE              ;je li nula
JE
                LOAD R1, (BRZ)          ;ako je povecaj brojac
                ADD R1, 1, R1
                STORE R1, (BRZ)
NIJE
                STORE R0, (BVJ)         ;salji podatak na BVJ
                STORE R0, (PVJ1IEND)    ;kraj obrade
VAN
                POP R0
                MOVE R0, SR
                POP R1
                POP R0
                RETI

OBR2
                STORE R0, (PVJ2IACK)    ;prihvat prekida
                LOAD R0, (BRZ)          ;ucitaj broj nula
                STORE R0, (PVJ2DR)      ;posalji ga VJ
                STORE R0, (PVJ2IEND)    ;kraj obrade prekida
                JR VAN
```

3.6.4 Sklopovi CT i PIO

Rješenje:

```
CTLR        `EQU 0FFFF0000
CTCR        `EQU 0FFFF0004
CTIACK      `EQU 0FFFF0008
CTIEND      `EQU 0FFFF000C
PIOUR       `EQU 0FFFF1000
PIODR       `EQU 0FFFF1004
PIOIACK     `EQU 0FFFF1008
PIOIEND     `EQU 0FFFF100C

            `ORG 0
            JR MAIN
            `ORG 8
            DW IRQ
            `ORG 0C
FIQ
            PUSH R0
            MOVE SR, R0
            PUSH R0

            STORE R0, (PIOIACK)     ;prihvat prekida
            LOAD R0, (PIODR)
            SHL R0, 8, R0           ;mnozenje s 256 , 2^8
            STORE R0, (CTLR)        ;nova konstanta za CT
```

```
                MOVE 0, R0
                STORE R0, (BROJAC)      ;reset brojaca
                STORE R0, (PIOEND)      ;kraj obrade

                POP R0
                MOVE R0, SR
                POP R0
                RETN

MAIN
                MOVE %B 10100000, SR    ;omoguci prekid na INT1

                MOVE  0, R0
                STORE R0, (CTLR)        ;upisuje se nula jer je  FFFF najvise sto stane u CTLR
                                        ;a kada je 0 to kao da pise 10000
                MOVE %B 011, R0
                STORE R0, (PIOUR)       ;ulazni nacin, prekid
                STORE R0, (CTCR)        ;brojilo broji, postavlja prekid
LOOP
                JR LOOP

BROJAC          DW 0

IRQ
                PUSH R0
                MOVE SR, R0
                PUSH R0

                STORE R0, (CTIACK)      ;prihvat prekida
                LOAD R0, (BROJAC)
                ADD R0, 1, R0           ;povecaj brojac
                STORE R0, (BROJAC)
                STORE R0, (CTIEND)      ;kraj prekida

                POP R0
                MOVE R0, SR
                POP R0
                RETI
```

3.6.6. Sklop CT, uvjetna vanjska jedinica i dvije bezuvjetne vanjske jedinice

Rješenje:

```
CTLR        `EQU 0FFFF0000
CTCR        `EQU 0FFFF0004
CTIACK      `EQU 0FFFF0008
CTIEND      `EQU 0FFFF000C
UVJDR       `EQU 0FFFF1000
UVJSB       `EQU 0FFFF1004
```

```
BVJ1        `EQU 0FFFF2000
BVJ2        `EQU 0FFFF3000

            `ORG 0
            MOVE 10000, SP
            JR MAIN
            `ORG 0C
CTFIQ
            PUSH R0
            STORE R0, (CTIACK)      ;...

            LOAD R0, (BROJAC)       ;slanje brojaca na BVJ2
            STORE R0, (BVJ2)
            MOVE 0, R0              ;reset brojaca
            STORE R0, (BROJAC)

            STORE R0, (CTIEND)      ;...
            POP R0
            RETN

MAIN
            MOVE %D 4000, R0        ;init CT
            STORE R0, (CTLR)
            MOVE %B 11, R0
            STORE R0, (CTCR)
LOOP
            LOAD R0, (UVJSB)        ;cekanje spremnosti
            OR R0, R0, R0
            JR_Z LOOP

            LOAD RO, (UVJDR)        ;citanje podatka
            AND R0, 1, R1
            JR_Z PARAN
NEPAR
            STORE R0, (BVJ1)        ;neparan broj na BVJ1
            STORE R0, (UVJSB)
            JR LOOP
PARAN
            LOAD R0, (BROJAC)       ;paran povecava brojac
            ADD R0, 1, R0
            STORE R0, (BROJAC)

            STORE R0, (UVJSB)
            JR LOOP

BROJAC DW 0
```

3.6.15. Sklopovi PIO i DMA te disk

Rješenje:

```
PIOUR       `EQU 0FFFF0000
PIODR       `EQU 0FFFF0004
PIOIACK     `EQU 0FFFF0008
PIOIEND     `EQU 0FFFF000C
DMASRC      `EQU 0FFFF2000
DMADST      `EQU 0FFFF2004
DMACNT      `EQU 0FFFF2008
DMAUR       `EQU 0FFFF200C
DMASTART    `EQU 0FFFF2010
DMAIACK     `EQU 0FFFF2014
DISC        `EQU 0FFFF4000

            `ORG 0
            MOVE 10000, SP
            JR MAIN
            `ORG 0C
PIOFIQ
            PUSH R0
            PUSH R1
            MOVE SR, R0
            PUSH R0

            STORE R0, (PIOIACK)
            LOAD R0, (PIODR)        ;R0 podatak
            LOAD R1, (ODMAK)        ;R1 odmak od adrese 800 ali u riječima ne bajtovima

            SHL R1, 2, R1           ;mnozimo s 4 iz navedenih razloga
            STORE R0, (R1 + 800)    ;spremamo
            SHR R1, 2, R1           ;vracamo natrag

            ADD R1, 1, R1           ;povecavamo odmak
            CMP R1, %D500           ;je li doslo do kraja
            JR_EQ KRAJ

            STORE R1, (ODMAK)       ;ako nije nista, vracamo se u main i spremamo odmak
VAN
            STORE R1, (PIOIEND)
            POP R0
            MOVE R0, SR
            POP R1
            POP R0
            RETN
KRAJ                                ;ako je...
            MOVE 0, R1              ;resetiram odmak
            STORE R1, (ODMAK)
INITDMA                             ;init DMA
            MOVE 800, R0
            STORE R0, (DMASRC)

            MOVE DISC, R0
```

```
                STORE R0, (DMADST)

                MOVE %D 500, R0
                STORE R0, (DMACNT)

                MOVE 0, R0
                STORE R0, (DMAUR)       ;dest mem, src mem, zaust proc, ne prekid
                STORE R0, (DMASTART)    ;pokreni
                STORE R0, (DMAIACK)     ;potvrda kraja
                JR VAN

MAIN
                MOVE %B 011, R0         ;init pio
                STORE R0, (PIOUR)
LOOP
                JR LOOP

ODMAK DW 0
```

3.6.16. Sklopovi PIO, CT i DMA te disk

Rješenje:

```
PIOUR       `EQU 0FFFF0000
PIODR       `EQU 0FFFF0004
PIOIACK     `EQU 0FFFF0008
PIOIEND     `EQU 0FFFF000C
DMASRC      `EQU 0FFFF4000
DMADST      `EQU 0FFFF4004
DMACNT      `EQU 0FFFF4008
DMAUR       `EQU 0FFFF400C
DMASTART    `EQU 0FFFF4010
DMAIACK     `EQU 0FFFF4014
DISC        `EQU 0FFFF6000
CTLR        `EQU 0FFFF8000
CTCR        `EQU 0FFFF8004
CTIACK      `EQU 0FFFF8008
CTIEND      `EQU 0FFFF800C

                `ORG 0
                MOVE 10000, SP          ;inicijalizacija stoga
                JR MAIN
                `ORG 8
                DW PIOIRQ
                `ORG 0C
CTFIQ                                   ;prekidni program CT-a
                PUSH R0                 ;spremanje konteksta
                PUSH R1
                MOVE SR, R0
```

```
            PUSH R0
            STORE R0, (CTIACK)          ;prihvat prekida
INITDMA
            MOVE DISC, R0               ;izvor
            STORE R0, (DMASRC)

            LOAD R0, (BRBLOKOVA)        ;broj blokova se ,mnozi s 80, 80 =
            SHL R0, 6, R1               ;64 (2^6) +
            SHL R0, 4, R0               ;16 (2^4)
            ADD R0, R1, R0
            ADD R0, %D 1000, R0         ;1000 pocetak podataka
            STORE R0, (DMADST)

            MOVE %D 20, R0              ;20 kolicina podataka
            STORE R0, (DMACNT)

            MOVE  0, R0
            STORE R0, (DMAUR)           ;dst mem, src mem, zaust proc, ne prekid
            STORE R0, (DMASTART)        ;pokreni
            STORE R0, (DMAIACK)         ;potvrdi

            LOAD R0, (BRBLOKOVA)        ;povecaj broj blokova
            ADD R0, 1, R0
            STORE R0, (BRBLOKOVA)
            CMP R0, %D 100              ;je li kraj
            JR_EQ GOTOVO

            STORE R0, (CTIEND)          ;ako nije povratak
            POP R0
            MOVE R0, SR
            POP R1
            POP R0
            RETN
GOTOVO                                  ;ako je zaustavljanje procesora
            STORE R0, (CTIEND)
            HALT

MAIN
            MOVE %B 10100000, SR        ;omoguci prekid na INT1
INITCT
            MOVE %D 10000, R0           ;konastanta za brojilo
            STORE R0, (CTLR)
            MOVE  %B 11, R0             ;broji i postavljaj prekid
            STORE R0, (CTUR)
INITPIO
            MOVE %B 010, R0             ;izlazni nacin, prekid
            STORE R0, (PIOUR)
LOOP
            JR LOOP

BRBLOKOVA   DW 0                        ;broj poslanih blokova
ODMAKPIO    DW 0                        ;odmak od adrese  %D 1000
```

```
PIOIRQ                                  ;prekidni program PIO-a
        PUSH R0
        PUSH R1
        MOVE SR, R0
        PUSH R0

        STORE R0, (PIOIACK)             ;prihvat prekida
        LOAD R1, (ODMAKPIO)
        LOAD R0, (R1 + %D 1000)         ;adresa= %D 1000 + odmak
        STORE R0, (PIODR)

        ADD R1, 4, R1
        STORE R1, (ODMAKPIO)            ;povecaj odmak
        STORE R0, (PIOIEND)

        POP R0
        MOVE R0, SR
        POP R1
        POP R0
        RETI
```