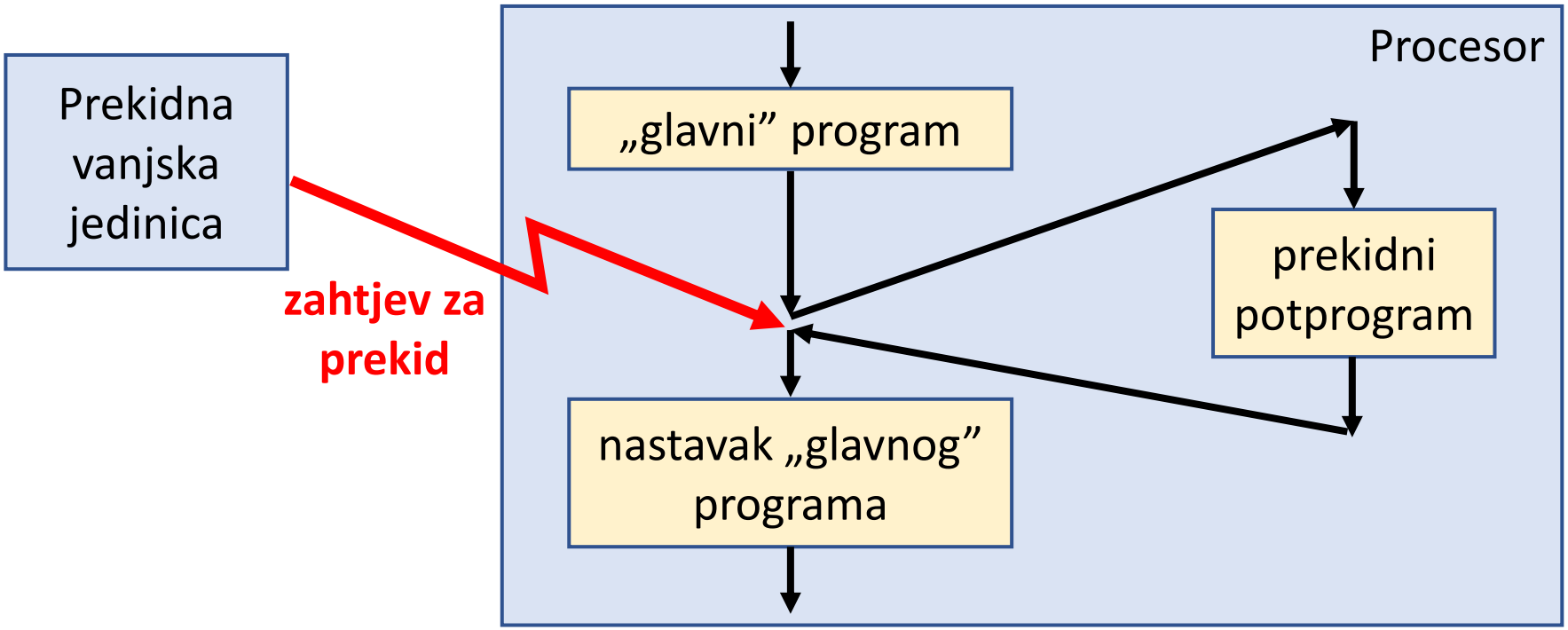
Prekidni UI prijenos

Podsjetnik

- Program prenosi podatke, ali ne inicira prijenos - to radi vanjska jedinica pomoću zahtjeva za prekid
- Prednosti:
  - nema gubitka/uvišestručenja podataka
  - nema čekanja na spremnost vanjske jedinice



Prihvaćanje zahtjeva za prekid

- Prekidni sustavi znatno se razlikuju od procesora do procesora, ali postoji određeno načelno ponašanje koje je zajedničko svima:

1. Procesor izvodi program, a VJ postavlja zahtjev za prekid (*interrupt request*) pomoću posebnog prekidnog priključka
2. Procesor izvodi trenutnu naredbu do kraja
   - tj. ispituje ima li zahtjeva za prekid tek na kraju izvođenja naredbe
3. Ako je u procesoru dozvoljeno prihvaćanje postavljenog prekida, onda procesor prihvaća zahtjev za prekid
   - u suprotnom nastavlja s izvođenjem glavnog programa
4. Prihvaćanje prekida sastoji se od:
   1. Procesor zabranjuje prihvaćanje daljnjih prekida (osim eventualno prekida jačeg prioriteta ako ih podržava)
   2. Procesor pohranjuje registar PC, a često i registar stanja (može pohranjivati i druge registre)
   3. Procesor određuje adresu prekidnog potprograma
   4. Procesor skače u prekidni potprogram

Prekidni potprogrami

- Prekidni potprogrami poslužuju prekidne vanjske jedinice.
- Organizacija prekidnog potprograma ovisi o vrsti, broju i prioritetima VJ-a, ali i o prekidnom sustavu procesora. Načelno, prekidni potprogrami rade ovako:

1. Spremanje konteksta
   - kontekst = svi registri koje potprogram mijenja, a nisu automatski spremljeni prilikom prihvaćanja prekida
2. Otkrivanje uzročnika prekida (ako ima više potencijalnih uzročnika)
   - tj. otkriva se koja VJ je izazvala prekid *
3. Dojavljivanje VJ da je **prekid prihvaćen**, VJ mora ukloniti zahtjev za prekid *
4. Posluživanje VJ
5. Obnavljanje konteksta
6. Ponovno dozvoljavanje prekida **
7. Dojavljivanje VJ da je njezin prekid obrađen (samo kod nekih procesora)
8. Izlazak iz prekidnog potprograma i povratak u glavni program na mjesto gdje je bio prekinut

---

* ovisno o procesoru može se izvesti sklopovski ili programski

** ovisno o procesoru može se izvesti prilikom obnove konteksta (5) ili kod izlaska iz prekidnog potprograma (8)

Prekidni sustav ARM-a

Prekidni priključci

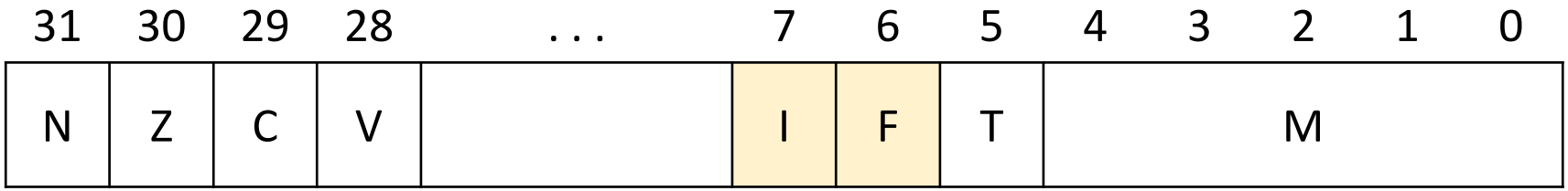
- ARM ima **dva prekidna priključka** aktivna visoko:
  - **IRQ** (*interrupt request*) je „obični prekid” nižeg prioriteta koji služi za obradu običnih zahtjeva za prekid
  - **FIQ** (fast interrupt request) je „brzi prekid” višeg prioriteta koji služi kad je potreban brzi odziv na prekid

- Prekidne vanjske jedinice mogu postavljati zahtjeve za prekid preko IRQ i FIQ (u ovisnosti na koji priključak su spojene)

- Prekidne linije su tipa spojeni-ILI što znači da će zahtjev s bilo koje jedinice izazvati zahtjev za prekid

Prekidni priključci

- Oba prekida su tzv. **maskirajući** prekidi (*maskable interrupts*) – mogu se programski maskirati, tj. onemogućiti i omogućiti (ili zabraniti i dozvoliti).
- Prekidima se upravlja pomoću bitova **I** i **F** u registru **CPSR**:
  - **onemogućuju** se upisom **1** u bit I odnosno F
  - **omogućuju** se upisom **0** u bit I odnosno F
  - inicijalno su zabranjeni pa ih se mora programski obrisati ukoliko se želi prihvaćati prekide



Prihvat prekida

- Postojanje prekida ispituje se **na kraju** svake naredbe
- Ispitivanje i prihvaćanje prekida ovisi o stanju zastavica **I** i **F** te o trenutačnim zahtjevima za prekid:
  - Ako je aktivan FIQ i zastavica F je 0, prihvaća se **FIQ**
  - Ako je aktivan IRQ i zastavica I je 0, prihvaća se **IRQ**
  - Ako se prekid ne prihvati (ili ako prekidi nisu aktivni), procesor nastavlja s izvođenjem programa
- Prekid se prihvaća tek nakon što se naredba koja je u fazi izvođenja izvede do kraja (zbog protočne strukture ovo nije tako jednostavno)
- Prihvaćeni prekid je **iznimka** koja se dalje obrađuje ovisno o vrsti prekida

Kontekst i stogovi

- Obavezno treba **spremiti** kontekst prekidnog potprograma
  - Registre PC i CPSR ne treba spremati jer se oni automatski spremaju na početku obrade iznimke u LR i SPSR
  - Za FIQ ne treba spremati R8-R12 jer postoje privatni registri R8_fiq-R12_fiq koji se koriste samo u obradi iznimke FIQ
  - Za sve načine rada*, pa tako i za IRQ i FIQ, postoje privatni registri R13 i R14
    - R14 treba spremati ako ga mijenjamo, jer je u njemu pohranjena adresa za povratak iz prekida
    - R13 ne treba spremati jer je on neovisan o R13 u glavnom programu
      - Međutim, zasebni registri R13 znači da svaki način rada ima vlastiti stog koji se koristi u tom načinu rada

* Osim usr i sys koji imaju zajedničke registre R13 i R14

Kontekst i stogovi

- Za sve načine rada koje ćemo koristiti u programu, **treba definirati zasebni stog**, tj. treba inicijalizirati registre R13_mod
  - u praksi je to uvijek stog za način *svc* te, po potrebi, za *irq* i/ili *fiq*
- Veličina stogova ovisit će o njihovom korištenju u obradi iznimaka
  - Mi ćemo zbog jednostavnosti rezervirati po 0x400 B za svaki stog, jer će to za naše potrebe biti sasvim dovoljno.

- Stogove definiramo tako da na početku obrade iznimke *Reset* redom prelazimo u pojedine načine rada i inicijaliziramo registar R13. Na kraju se opet vraćamo u način *Supervisor* i pokrećemo glavni program.

- Konkretni programi bit će pokazani u primjerima s konkretnim vanjskim jedinicama
  - Sada ćemo pokazati samo „kostur programa” (vidi sljedeće slajdove)...

Inicijalizacija obrade iznimaka


```
        ORG   0
    B     POSLUZI_RESET     ; skok na obradu iznimke RESET
    ORG   0x18
    B     POSLUZI_IRQ       ; skok na obradu iznimke IRQ
    ORG   0x1C
    ; obrada iznimke FIQ (bez skoka)
POSLUZI_FIQ  ...
        ...
        SUBS   PC, LR, #4       ; povratak u glavni program
```

Napomene:

- Potprogram za FIQ može se izravno napisati na adresi 0x1C jer iza te adrese nema drugih adresa za obradu iznimaka.
- Ako potprogram za FIQ poziva drugi potprogram treba na početku spremiti LR na stog/u registar.
- Potprogram za FIQ može slobodno mijenjati registre R8-R12 (jer su oni jedinstveni registri za način rada *fiq*). Ako koristi ostale registre treba ih spremiti kao kontekst na stog.

Inicijalizacija obrade iznimaka


```
POSLUZI_IRQ  ; obrada iznimke IRQ
        STMFD SP!, {R0,R1,R8}    ; spremanje konteksta
              ...
        LDMFD SP!, {R0,R1,R8}
        SUBS   PC, LR, #4        ; povratak u glavni program
```

Napomene:

- Potprogram za IRQ ne može se izravno napisati na adresi 0x18 jer je odmah iza na 0x1C adresa za obradu iznimke FIQ
- Ako potprogram za IRQ koristi neke registre ili poziva drugi potprogram treba te registre i LR spremati na stog
- Potprogram za IRQ može slobodno mijenjati registre nakon spremanja konteksta

Inicijalizacija obrade iznimaka


```
POSLUZI_RESET ; obrada iznimke RESET
        ; Inicijalizacija stogova
        ; stog za IRQ
        MSR   CPSR, #0b11010010  ; prelazak u način rada IRQ
        MOV   R13,  #0x10000     ; inicijalizacija R13_irq

        ; stog za FIQ
        MSR   CPSR, #0b11010001  ; prelazak u način rada FIQ
        MOV   R13,  #0xFC00      ; inicijalizacija R13_fiq

        ; stog za SVC
        MSR   CPSR, #0b11010011  ; prelazak u način rada SVC
        MOV   R13,  #0xF800      ; inicijalizacija R13_svc
```

(nastavak na sljedećem slajdu)

Napomene:

- 110 je inicijalna vrijednost bitova nakon RESET (koju ne želimo promijeniti!)

Inicijalizacija obrade iznimaka

(nastavak s prethodnog slajda)

```
        ; inicijalizacija prekidnih vanjskih jedinica
        ...

        ; dozvoljavanje prekida IRQ i FIQ (bitovi 7 i 6 u CPSR-u) ako je potrebno
        MRS  R0, CPSR
        BIC  R0, R0, #0b11000000  ; brisanje bitova 7 i 6
        MSR  CPSR, R0

GLAVNI  B   GLAVNI          ; „koristan posao”
```

Napomene:

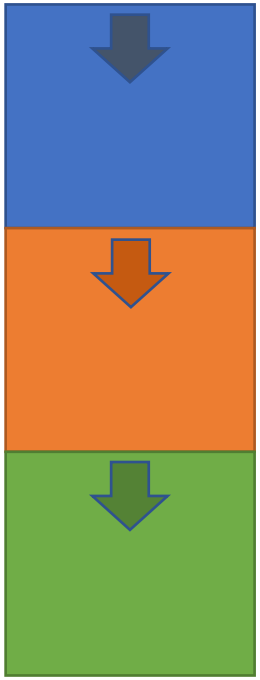
- Sekvenca MRS-BIC-MSR je standardna za promjenu pojedinih bitova u CPSR jer se ne smije nehotice mijenjati druge bitove.

Stogovi nakon incijalizacije

IRQ stog: 0x10000

FIQ stog: 0xFC00

SVC stog: 0xF800



U ovom primjeru inicijalizacije smo pretpostavili da program neće trebati više od 0x400 bajtova na pojedinom stogu

Jedinica RTC

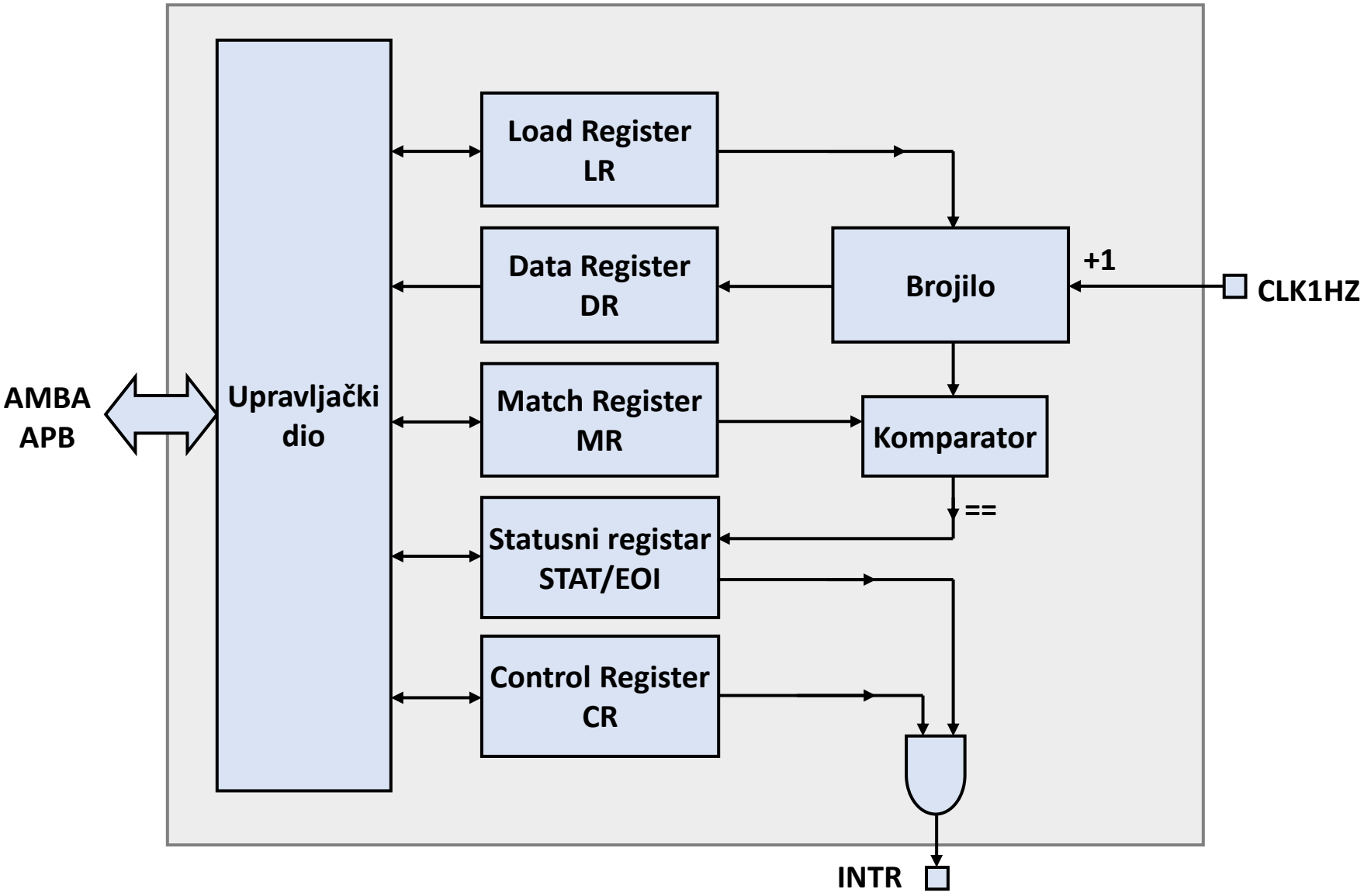
RTC kao uvjetna i prekidna jedinica

RTC - uvod

- **RTC** (Real Time Clock) je jedinica kojom se može prebrajati zadani broj impulsa koji dolaze na **ulazni priključak CLK1HZ**
  - ako su impulsi pravilne frekvencije, RTC služi kao mjerač vremena
  - ako se radi o nekim općim impulsima, RTC služi kao brojač impulsa

- RTC-u se zadaje broj impulsa koje treba prebrojiti. Zadani broj impulsa pamti se u registru **MR** (Match Register)
- RTC ima 32-bitno brojilo koje se uvećava na svaki impuls na **priključku CLK1HZ**
- Kada **brojilo postane jednako MR-u** onda
  → RTC postaje **spreman** i postavlja zahtjev za prekid (ako je tako zadano u upravljačkom registru **CR**)

RTC - građa


AMBA
APB
Upravljački
dio
Load Register
LR
Data Register
DR
Match Register
MR
Statusni registar
STAT/EOI
Control Register
CR
Brojilo
+1
CLK1HZ
Komparator
==
INTR

RTC - građa

- Dijelovi* RTC-a su:
  - Upravljački dio (upravljanje, APB sučelje, pristup registrima)
  - Registri (LR**, DR, MR, STAT/EOI, CR)
  - Brojilo
  - Komparator

- ARM preko APB sabirnice pristupa RTC-ovim registrima i tako komunicira s RTC-om

---

* Većina naziva (imena registara, signala, razne kratice itd.) u RTC-u počinje prefiksom RTC čime ti nazivi postaju nepregledni i predugački. Gdje god možemo ispuštamo ovaj prefiks i po potrebi dodajemo podcrte da imena budu preglednija.

** Ponašanje registra LR je ovdje malo pojednostavljeno

Registri DR, LR i MR

- Registar **DR** – Data Register (može se samo čitati)
  - 32-bitni registar - čitanje brojila
  - Operacija čitanja vraća trenutačnu vrijednost brojila (ne prekorisno)
    - rijetko se koristi, jedino za provjeru dokle je brojilo došlo

- Registar **LR** – Load Register
  - 32-bitni registar - upis u brojilo
  - Operacija pisanja upisuje podatak u LR i kopira ga u brojilo
    - služi za inicijalizaciju brojila
  - Operacija čitanja vraća zadnju upisanu vrijednost u LR (ne prekorisno)

- Registar **MR** – Match Register
  - 32-bitni registar
  - Operacija pisanja upisuje podatak u MR
    - služi za zadavanje ciklusa brojenja, tj. konstante brojenja
  - Operacija čitanja vraća zadnju upisanu vrijednost u MR (ne prekorisno)

Registri STAT/EOI i CR

- Statusni registar **STAT/EOI** – Status/End Of Interrupt
  - 1-bitni registar kod čitanja (najniži bit), 0-bitni kod pisanja!
  - Operacija čitanja vraća trenutačno stanje RTC-a.
    Stanje 1 označava spremnost RTC-a, tj. da je dovršen ciklus brojenja
  - Operacija pisanja briše ovaj registar, bez obzira koji podatak se upiše (poslani podatak se zanemaruje)

- Registar **CR** – Control Register
  - 1-bitni registar (najniži bit, ostali su nedefinirani)
  - Operacija pisanja upisuje podatak u CR i služi za dozvoljavanje ili zabranjivanje prekida
    - **0** znači da **RTC neće postavljati prekide**
    - **1** znači da će **RTC postavljati prekide kad postane spreman**
  - Operacija čitanja vraća zadnju upisanu vrijednost iz CR

Ponašanje brojila i RTC-a

- Ponašanje brojila:
  - Brojilo se **povećava** na svaki ulazni impuls
  - inicijalna vrijednost brojila je 0
  - Brojilo počinje brojati odmah nakon početka rada sustava !!
  - ako brojilo ima vrijednost 0xFFFFFFFF, nakon sljedećeg impulsa ponovno postaje 0
  - Kada **brojilo postane jednako MR-u**, onda je **dovršen jedan ciklus brojenja**

- Ponašanje RTC-a na kraju ciklusa brojenja:
  - Brojilo i dalje nastavlja s brojenjem impulsa (postaje MR+1, pa MR+2 itd.)
  - RTC postaje spreman (STAT/EOI postaje 1)
  - Ako je zadano da treba postaviti prekid (CR je jednak 1), onda se aktivira INTR (izlazni priključak za postavljanje zahtjeva za prekid)

RTC - adrese

Adresa	Naziv registra	Opis
bazna_adr + 0	DR	32-bitni Data Register (može se samo čitati)
bazna_adr + 4	MR	32-bitni Match Register
bazna_adr + 8	STAT/EOI	1/0-bitni Statusni registar (Status/Interrupt Clear)
bazna_adr + C	LR	32-bitni Load Register
bazna_adr + 10	CR	1-bitni Control register

- Inicijalno, svi registri RTC-a su 0

Inicijalizacija RTC-a

- Prije rada s RTC-om treba mu **inicijalizirati brojilo** (pomoću LR-a) i **zadati konstantu brojanja** u MR
  - U brojilo se upisuje **0**
  - U MR se upisuje broj impulsa koji treba odbrojiti

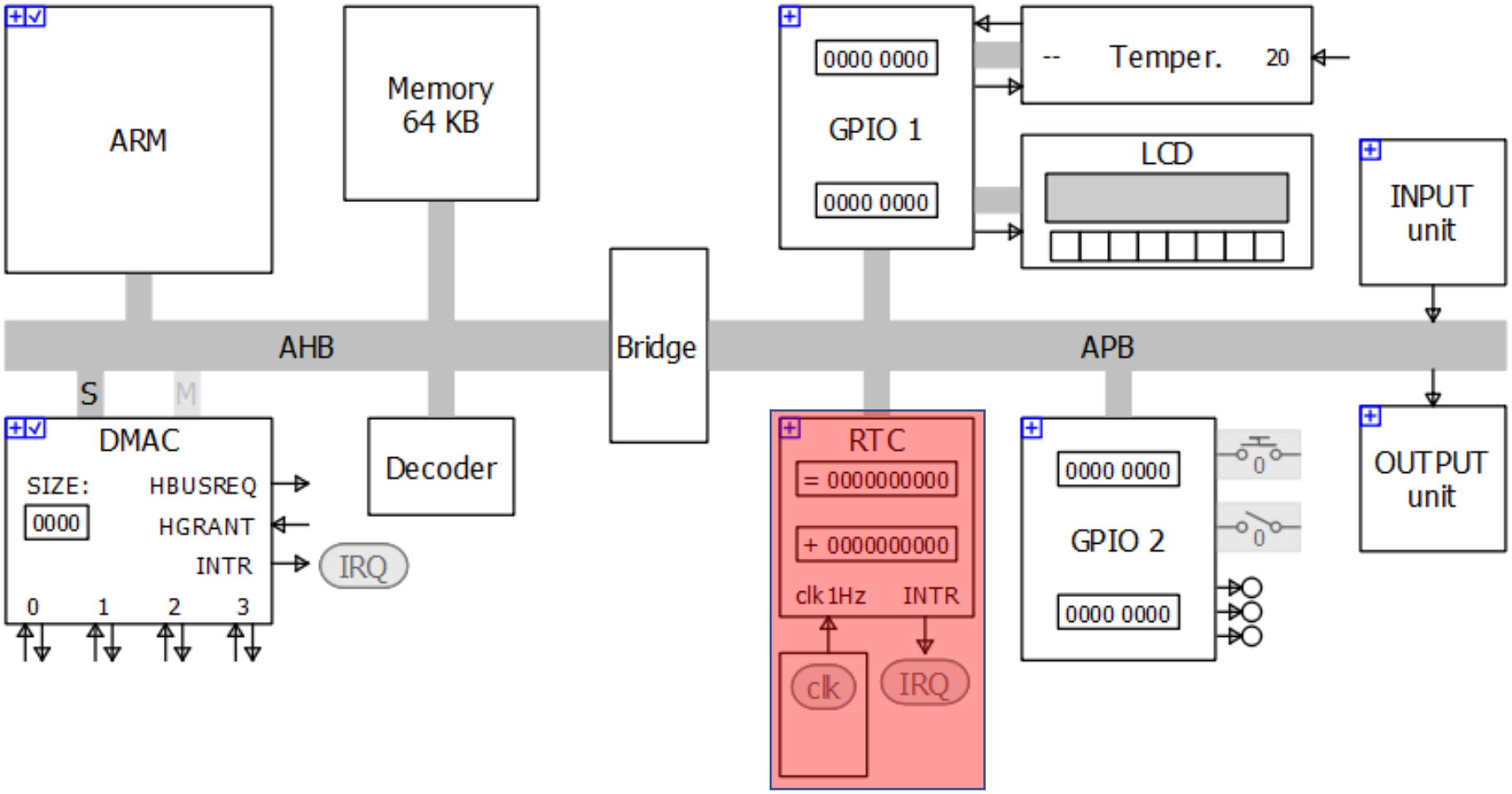
  - Alternativni način:
  - Pročitati iz DR trenutnu vrijednost brojila
  - U MR upisati trenutnu vrijednost brojila + broj impulsa koji treba odbrojiti

- **Ako se koristi prekid** treba upisati 0 ili 1 u CR
  - Tipično se upisuje 1 tako da RTC generira prekide
  - Ako se zada 0, onda će RTC efektivno raditi u uvjetnom načinu (što najčešće nema previše smisla)

Programiranje RTC-a

- Kad RTC postane **spreman** što ARM prepoznaje prekidom (ili uvjetno ispitivanjem registra STAT/EOI) treba ga **poslužiti**:
  - **Obavezno obrisati stanje spremnosti u STAT/EOI** (posljedično će RTC ukloniti zahtjev za prekid)
  - Ako se želi **zaustaviti brojenje**, treba u CR upisati 0 tako da više ne postavlja prekide (ako se RTC poslužuje uvjetno, onda se jednostavno prestane ispitivati njegova spremnost, a u CR-u je ionako već upisana 0 od prije)
  - Ako se želi **nastaviti s brojenjem**, onda treba:
    - obavezno ponovno inicijalizirati ciklus brojanja (dva koraka s prethodnog slajda)

RTC u SSPARCSS-u



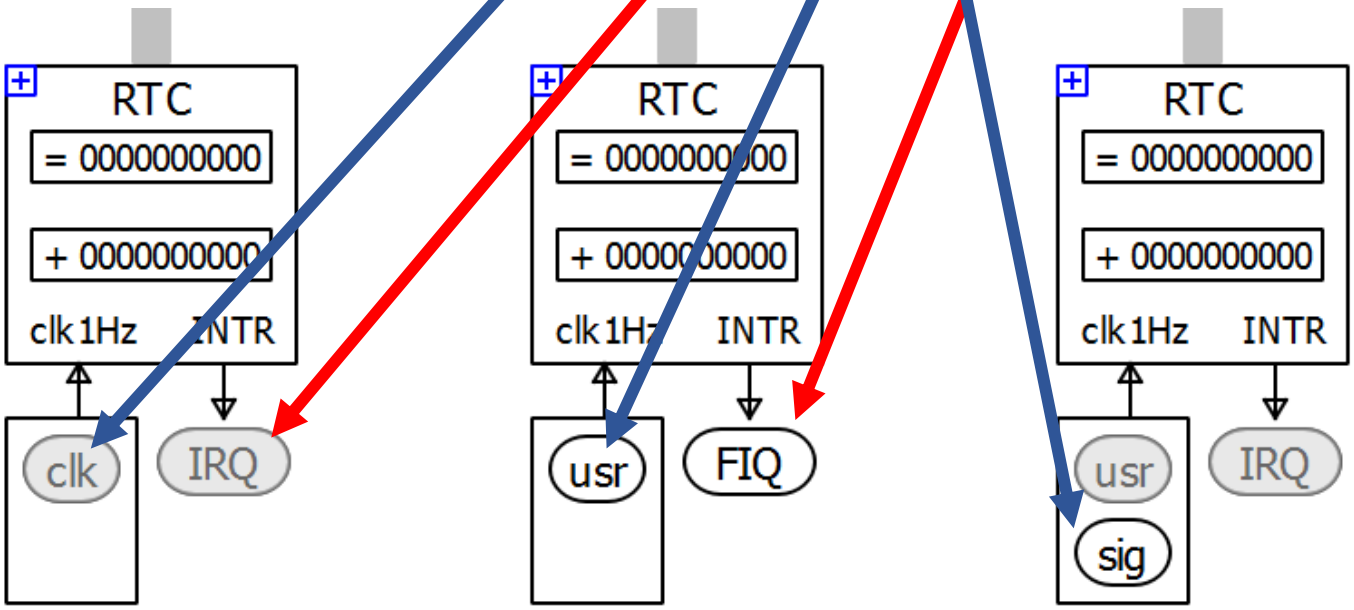
Bazne adrese vanjskih jedinica:

GPIO1 = FFFF 0F00 RTC = FFFF 0E00 INPUT = FFFF 0D00

GPIO2 = FFFF 0B00 OUTPUT = FFFF 0C00

RTC u SSPARCSS-u

- Nakon učitavanja exe-fajla, ali **prije pokretanja simulacije** može se birati:
  1. na koju će prekidnu liniju biti spojen RTC (IRQ ili FIQ)
  2. što će biti izvor ulaznog impulsa:
     - signal konstantne frekvencije (clk) s periodom 25 clock-ova
     - ili impulse koje će zadavati korisnik (usr) gumbom sig (gumb sig je dostupan samo dok traje simulacija)



RTC - uvjetno brojenje impulsa - primjer

RTC je na adresi FFFF 0E00. Na ulaz RTC-a spojen je stroj koji proizvodi vijke. Na svaki proizvedeni vijak generira se impuls.

Napisati program koji treba beskonačno prebrajati pakete vijaka i osvježavati njihov broj u lokaciji BRPAK. U jednom paketu ima 20 vijaka.

Zadatak treba riješiti uvjetnim prijenosom, odnosno ispitivanjem spremnosti RTC-a. (Ovo rješenje će biti neefikasno)

RTC - uvjetno brojenje impulsa - primjer


```
    ORG   0
INIT  LDR   R0, RTC        ; dohvati adresu RTC-a

    MOV   R1, #0
    STR   R1, [R0, #0x0C] ; pobrisati brojilo preko LR

    MOV   R2, #20         ; broj vijaka u jednom paketu
    STR   R2, [R0, #0x04] ; upisati konstantu brojenja u MR

    STR   R1, [R0, #0x10] ; zabraniti prekid RTC-u u CR (nije nužno)
```

(nastavak na sljedećem slajdu)

RTC - uvjetno brojenje impulsa - primjer


```
GLAVNI ; glavni program samo prebraja pakete

CEKAJ  LDR  R1, [R0, #0x08]  ; dohvatiti status RTC-a
       TST  R1, #1           ; ispitati najniži bit
       BEQ  CEKAJ            ; dok je status==0 => čekaj spremnost

PAKET  ; spreman => odbrojeno je 20 impulsa
       MOV  R1, #0           ; inicijaliziraj sljedeći ciklus brojenja
       STR  R1, [R0, #0x0C]  ; pobrisati brojilo (MR ostaje 20)

       STR  R1, [R0, #0x08]  ; obrisati status (šalje se bilo što)

       LDR  R1, BRPAK        ; povećati brojač paketa BRPAK
       ADD  R1, R1, #1
       STR  R1, BRPAK

       B    CEKAJ            ; beskonačno prebrajaj pakete



RTC   DW    0xFFFF0E00       ; adresa RTC-a
BRPAK DW    0                ; brojač proizvedenih paketa
```

RTC - prekidno brojenje impulsa - primjer

Kao prethodni zadatak, ali RTC je spojen na IRQ i zadatak treba riješiti prekidnim prijenosom:

RTC je na adresi FFFF 0E00 i spojen je na IRQ. Na ulaz RTC-a spojen je stroj koji proizvodi vijke. Na svaki proizvedeni vijak generira se impuls.

Napisati program koji treba beskonačno prebrajati pakete vijaka i osvježavati njihov broj u lokaciji BRPAK. U jednom paketu ima 20 vijaka.

RTC - prekidno brojenje impulsa - primjer


```
    ORG   0
    B     MAIN              ; skok na obradu iznimke RESET

    ORG   0x18
    B     PAKET             ; skok na obradu iznimke IRQ


MAIN   ; Inicijalizacija stogova

    ; stog za način irq
    MSR   CPSR, #0b11010010   ; prelazak u način rada IRQ
    MOV   R13,  #0x10000      ; inicijalizacija R13_irq

    ; stog za način supervisor
    MSR   CPSR, #0b11010011   ; prelazak u način rada SVC
    MOV   R13,  #0xFC00       ; inicijalizacija R13_svc
```

(nastavak na sljedećem slajdu)

RTC - prekidno brojenje impulsa - primjer

(nastavak s prethodnog slajda)

```
    ; inicijalizacija RTC-a
    LDR   R0, RTC        ; dohvati adresu RTC-a

    MOV   R1, #0
    STR   R1, [R0, #0x0C] ; pobriši brojilo

    MOV   R1, #20         ; broj vijaka u jednom paketu
    STR   R1, [R0, #0x04] ; upiši konstantu brojenja u MR

    MOV   R1, #1
    STR   R1, [R0, #0x10] ; dozvoli prekid RTC-u (obavezno)

    MRS   R0, CPSR        ; dozvoli prekid IRQ
    BIC   R0, R0, #0b10000000
    MSR   CPSR, R0

RADI   B     RADI   ; neki koristan posao u nastavku glavnog programa


RTC    DW    0xFFFF0E00      ; adresa RTC-a
BRPAK  DW    0               ; brojač proizvedenih paketa
```

RTC - prekidno brojenje impulsa - primjer


```
PAKET ; Prekidni potprogram za IRQ - posluživanje RTC-a

    STMFD SP!, {R0-R1}     ; spremi kontekst

    LDR   R0, RTC          ; dohvati adresu RTC-a

    MOV   R1, #0           ; inicijaliziraj sljedeći ciklus brojenja
    STR   R1, [R0, #0x0C]  ; pobriši brojilo (MR ostaje 20)

    STR   R1, [R0, #0x08]  ; obriši status (šalje se bilo što)

    LDR   R1, BRPAK        ; povećaj brojač paketa BRPAK
    ADD   R1, R1, #1
    STR   R1, BRPAK

    LDMFD SP!, {R0-R1}     ; obnovi kontekst
    SUBS  PC, LR, #4       ; povratak u glavni program
```

RTC - prekidno brojenje impulsa - primjer

- Iako jednostavnije i kraće, uvjetno rješenje je puno neefikasnije jer glavni program ne može raditi nikakav koristan posao, osim čekati da RTC postane spreman

- Prekidno rješenje omogućuje glavnom programu nesmetan rad većinu vremena (pretpostavka je da se 20 vijaka proizvodi cca minutu)

- Zadatak: Riješite zadatak tako da je RTC spojen na FIQ i tako da iskoristite prednosti FIQ-a u pogledu bržeg odziva na zahtjev za prekid.

RTC - mjerenje vremena - primjer

RTC je na adresi FFFF 0E00 i spojen je na FIQ. Na ulaz RTC-a spojen je generator impulsa frekvencije 10 kHz.

Napisati program koji treba svake 2 sekunde* pozvati potprogram TICK, a inače glavni program stalno poziva neki potprogram POSAO.

Potprograme TICK i POSAO ne treba pisati.

* KOMENTAR ZA SIMULATOR: u simulatoru se simulirano vrijeme mjeri brojem odsimuliranih ciklusa clock-a, a ne sekundama. Zato ovakve zadatke **ne možemo simulirati**, a također ne možemo simulirati ni odsječke programa određenog trajanja. Osim toga, trajanje izvođenja pojedinih naredaba i **pipeline nisu implementirani** u postojećem modelu ARM-a.

RTC - mjerenje vremena - primjer

Rješenje - izračun konstante:

ulaz: 10 kHz

trajanje brojenja: 2 sekunde

1. način: konstanta je umnožak ulazne frekvencije i željenog vremena koje RTC treba mjeriti:

10 kHz * 2 s = 10000 Hz * 2 s = 20000

2. način: „Ulaznu" frekvenciju 10 kHz treba pretvoriti u „izlaznu" frekvenciju 0,5 Hz. Potrebno je ostvariti „dijeljenje frekvencije":

10 kHz / 0,5 Hz = 10000 Hz / 0,5 Hz = 20000

RTC - mjerenje vremena - primjer


```
    ORG   0
    B     MAIN          ; skok na obradu iznimke RESET


    ORG   0x1C          ; obrada iznimke FIQ
FIQ   STMFD SP!, {LR}     ; pohrani kontekst (jer se poziva TICK)
    LDR   R8, RTC       ; dohvati adresu RTC-a

REINIT MOV   R9, #0       ; inicijaliziraj sljedeći ciklus brojenja
    STR   R9, [R8, #0x0C] ; pobrisati brojilo (MR ostaje 20000)

    STR   R9, [R8, #0x08] ; obrisati status (šalje se bilo što)

    BL    TICK          ; pozvati zadani potprogram


    LDMFD SP!, {LR}     ; obnovi kontekst
BACK  SUBS  PC, LR, #4    ; povratak u glavni program
```

RTC - mjerenje vremena - primjer


```
MAIN   ; Inicijalizacija stogova

STACKS MSR   CPSR, #0b11010001  ; prelazak u način rada FIQ
       MOV   R13,  #0x10000     ; inicijalizacija R13_fiq

       MSR   CPSR, #0b11010011  ; prelazak u način rada SVC
       MOV   R13,  #0xFC00      ; inicijalizacija R13_svc


INIT   BL    INIT_RTC          ; inicijalizacija RTC-a


ION    MRS   R0, CPSR          ; dozvoljavanje prekida FIQ
       BIC   R0, R0, #0b01000000
       MSR   CPSR, R0


RADI   BL    POSAO             ; koristan posao glavnog programa
       B     RADI
```

RTC - mjerenje vremena - primjer


```
INIT_RTC ; Inicijalizacija RTC-a
        STMFD  SP!, {R0-R1}
        LDR   R0, RTC        ; dohvati adresu RTC-a

        MOV   R1, #0
        STR   R1, [R0, #0x0C] ; pobrisati brojilo

        LDR   R1, KONST
        STR   R1, [R0, #0x04] ; upisati konstantu brojenja u MR

        MOV   R1, #1
        STR   R1, [R0, #0x10] ; dozvoliti prekid RTC-u (obavezno)

        LDMFD  SP!, {R0-R1}
        MOV    PC, LR

RTC     DW    0xFFFF0E00      ; adresa RTC-a
KONST   DW    20000           ; konstanta brojenja (ne stane u MOV)
```

Primjer - dva RTC-a u bezuvjetnom i prekidnom načinu

RTC1 je spojen na IRQ. RTC1 mjeri vremenske intervale od 0,1 sekunde a njegova ulazna frekvencija clk signala je 1 kHz.

RTC2 broji impulse koji dolaze s motora (svaki okretaj je jedan impuls).

Program treba svake desetinke sekunde izmjeriti brzinu motora izraženu kao broj okretaja u sekundi. Brzine treba spremati kao bajtove u memoriju u blok BRZINE. Nakon 1000 mjerenja treba zaustaviti program.

Idejno rješenje:

- Kad dođe prekid nakon 0,1 sekunde, onda treba očitati stanje brojila na RTC2 i od njega oduzeti prethodno stanje brojila. Ova razlika je broj okretaja u protekloj desetinki.
- RTC2 ne generira prekide, niti ga poslužujemo uvjetno niti mu zadajemo MR.

Primjer - dva RTC-a u bezuvjetnom i prekidnom načinu


```
    ORG  0
    B    GLAVNI

    ORG  0x18            ; adresa prekidnog potprograma
    B    MJERI


GLAVNI ; inicijaliziraj stogove
    MSR  CPSR, #0b11010010  ; prelazak u način rada IRQ
    MOV  R13,  #0x10000     ; inicijalizacija R13_irq

    MSR  CPSR, #0b11010011  ; prelazak u način rada SVC
    MOV  R13, #0xFC00       ; inicijalizacija R13_svc

    BL   UI_INIT            ; inicijalizacija oba RTC-a

    MRS  R0, CPSR           ; omogućiti prekid IRQ
    BIC  R0, R0, #0x80
    MSR  CPSR, R0
```

(nastavak na sljedećem slajdu)

Primjer - dva RTC-a u bezuvjetnom i prekidnom načinu

(nastavak s prethodnog slajda)

```
      MOV  R10, 0x1000
PETLJA LDR  R0, OFSET
      CMP  R0, R10
      BEQ  KRAJ
      B    PETLJA

KRAJ   SWI  0x123456



      ; proizvoljno odabrane bazne adrese (jer nisu zadane)
RTC1   DW   0xFFFF1000
RTC2   DW   0xFFFF2000

LAST   DW   0     ; zadnja vrijednost brojača okretaja
OFSET  DW   0     ; ofset u bloku za spremanje brzine

BRZINE DS   1000  ; brzine su bajtovi
```

Primjer - dva RTC-a u bezuvjetnom i prekidnom načinu


```
UI_INIT ; pomoćni potprogram za inicijalizaciju RTC-ova
        STMFD SP!, {R1, R2, R3}

        LDR   R1, RTC1        ; R1 = bazna adresa RTC1
        LDR   R2, RTC2        ; R2 = bazna adresa RTC2

        MOV   R3, #0          ; LR - brisanje brojila u oba RTC-a
        STR   R3, [R1, #0C]
        STR   R3, [R2, #0C]

        MOV   R3, #100
        STR   R3, [R1, #4]    ; MR - konstanta samo za RTC1

        MOV   R3, #1
        STR   R3, [R1, #10]   ; CR - prekidanje samo za RTC1

        LDMFD SP!, {R1, R2, R3}
        MOV   PC, LR
```

Primjer - dva RTC-a u bezuvjetnom i prekidnom načinu


```
MJERI ; obrada prekida nakon isteka jedne desetinke

    STMFD SP!, {R0-R2}

    LDR   R1, RTC1          ; dohvat adresa od RTC-ova
    LDR   R2, RTC2

    STR   R0, [R1, #8]      ; dojava prihvata prekida u RTC1
    MOV   R0, #0
    STR   R0, [R1, #0xC]    ; LR - brisanje brojila u RTC1
```

(nastavak na sljedećem slajdu)

Primjer - dva RTC-a u bezuvjetnom i prekidnom načinu

(nastavak s prethodnog slajda)

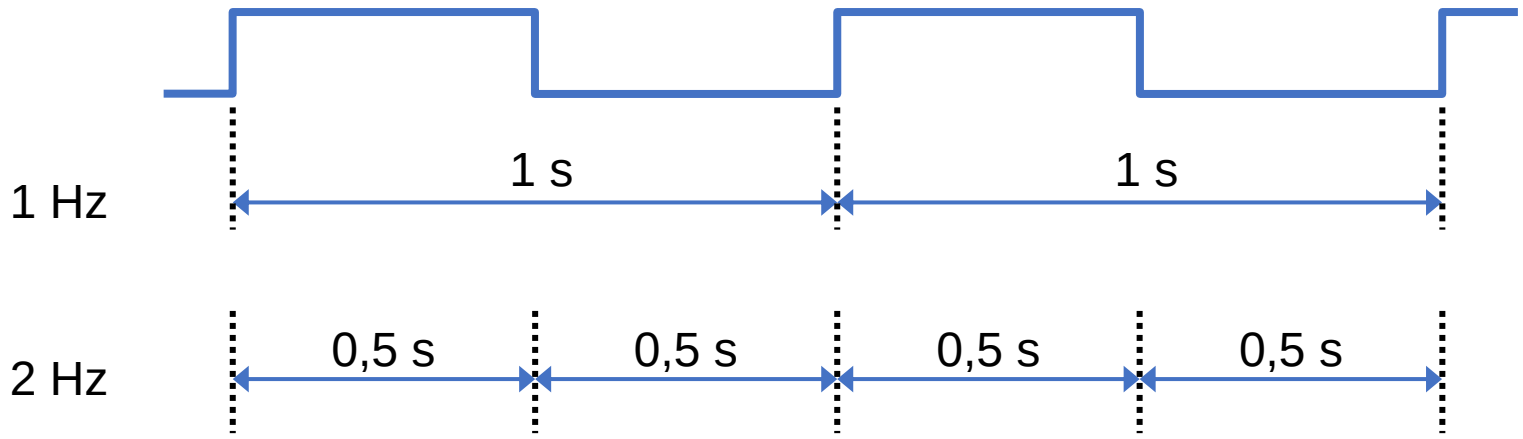
```
LDR   R0, LAST          ; R0 = prethodno stanje brojila
LDR   R1, [R2, #0]       ; R1 = DR (tj. stanje brojila od RTC2)
STR   R1, LAST          ; ažuriraj prethodno stanje brojila

SUB   R2, R1, R0         ; R2 = razlika broja okretaja
                  ;     u zadnjoj desetinki
ADD   R2, R2, R2, LSL #2  ; R2*10 (pretvorba okretaji/desetinka
ADD   R2, R2, R2          ;       u okretaji/sekunda)

MOV   R0, #BRZINE        ; dohvati baznu adresu i ofset
LDR   R1, OFSET
STRB  R2, [R0, R1]       ; spremi brzinu na adresu [BRZINA + OFSET]
ADD   R1, R1, #1
STR   R1, OFSET          ; povećaj OFSET

LDMFD SP!, {R0-R2}
SUBS  PC, LR, #4
```

Kombinirani primjeri GPIO i RTC

Primjer - generiranje impulsa

RTC je na adresi 0xFFFF0E00, a GPIO je na adresi 0xFFFF0B00.
RTC je spojen na FIQ. Na ulaz RTC-a spojen je signal od 20 Hz.

Potrebno je na LED koja je spojena na priključku XPA[5] od jedinice GPIO2 generirati pravokutni signal frekvencije 1 Hz.



Treba izračunati konstantu da se dobije dijeljenje frekvencije s 20 Hz na 2 Hz:

$$\frac{20}{\text{konstanta}} = 2 \quad \Rightarrow \quad \text{konstanta} = \frac{20}{2} = 10$$

Primjer - generiranje impulsa


```
    ORG  0
    B    GLAVNI

    ORG  0x1C          ; Adresa prekidnog potprograma za FIQ

    ; ne spremamo kontekst jer koristimo registre od načina FIQ
FIQ   LDR  R8, RTC       ; R8 je bazna adresa RTC-a
    LDR  R9, GPIO      ; R9 je bazna adresa GPIO-a

    STR  R10, [R8, #8]   ; prihvat prekida RTC-a
    MOV  R10, #0       ; reinicijalizacija RTC-a
    STR  R10, [R8, #0xC]

    ; promjena stanja pravokutnog signala na GPIO-u
SIGNAL LDRB R10, [R9]      ; dohvati prethodno stanje signala
    EOR  R10, R10, #0x20 ; invertiraj stanje
    STRB R10, [R9]      ; upiši novo stanje signala

    SUBS PC, LR, #4     ; povratak iz obrade prekida
```

Primjer - generiranje impulsa


```
GLAVNI ; glavni program

     ; stogovi se ne inicijaliziraju jer se ne koriste

     ; inicijalizacija RTC-a
RTC_I  LDR  R1, RTC        ; R1 je bazna adresa RTC-a
     MOV  R0, #10          ; konstanta brojenja
     STR  R0, [R1, #4]     ; upis u MR
     MOV  R0, #1           ; dozvoli RTC-u da zahtijeva prekid
     STR  R0, [R1, #0x10]  ; upis u CR
     MOV  R0, #0           ; briši brojilo u RTC-u
     STR  R0, [R1, #0xC]   ; upis u LR
```

(nastavak na sljedećem slajdu)

Primjer - generiranje impulsa

(nastavak s prethodnog slajda)

```
    ; inicijalizacija GPIO-vog porta A - XPA[0] je izlazni
GPIO_I LDR  R1, GPIO        ; R1 je bazna adresa GPIO-a

    MOV  R0, #0b00000001
    STRB R0, [R1, #8]

    ; omogućavanje prekida FIQ
    MRS  R0, CPSR
    BIC  R0, R0, #0x40
    MSR  CPSR, R0

PETLJA B    PETLJA          ; „koristan posao”


RTC    DW   0xFFFF0E00      ; bazne adrese RTC-a i GPIO-a
GPIO   DW   0xFFFF0B00
```

Primjer - generiranje impulsa

Komentari:

- U ovom zadatku se RTC poslužuje prekidnim načinom, a GPIO bezuvjetnim.
- Umjesto da pamtimo trenutačno stanje impulsa, jednostavno se pročita prethodno stanje s bita XPA[5]

Zadatak za vježbu:

- Ostvarite semafor. Neka crveno svjetlo traje 10s, žuto 1s a zeleno 5s.

Primjer - prikaz opsega temperature

GPIO1 preko vrata A prima temperaturu nekog procesa u Celzijevim stupnjevima. Temperaturni sklop je kao u ranijim primjerima.

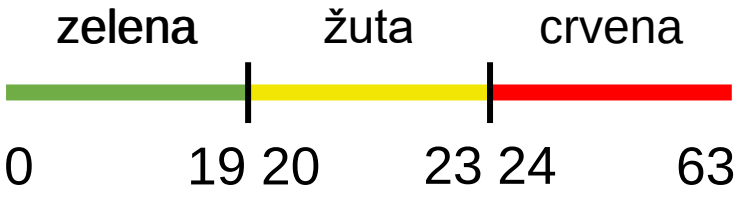
Pinovi 5-7 na GPIO2 vrata A služe za signalizaciju opsega temperature. Na njih su spojene crvena, žuta i zelena LED kao u prijašnjim primjerima.

Ako je temperatura veća od 23 °C, treba upaliti crvenu lampicu, a ugasiti žutu i zelenu. Isto treba učiniti sa žutom lampicom, ako je temperatura u rasponu od 20°C do 23 °C, odnosno sa zelenom ako je temperatura manja od 20 °C.

Provjeru temperature treba obavljati svake sekunde na sljedeći način:

- Provjeriti da li temperaturni sklop ima novu izmjerenu temperaturu
  - Ako ima, pročitati ju i osvježiti stanja LED prema novoj temperaturi
  - Ako nema, ostaviti trenutno stanje LED i nastaviti rad

Na RTC je spojen signal frekvencije 20 Hz i FIQ.



Primjer - prikaz opsega temperature

Komentari:

- U ovom zadatku imamo kombinaciju
  - RTC se poslužuje prekidnim načinom
  - Termometar je u uvjetnom načinu preko GPIO
  - LED su u bezuvjetnom preko GPIO

- S obzirom da prekid od RTC i spremnost Temperaturnog sklopa dolaze potpuno neovisno moramo se pobrinuti da sustav ispravno funkcionira
  - Prekid sa RTC se mora UVIJEK obraditi
  - Unutar obrade prekida s RTC provjeravamo spremnost Temp ali ako nije spremna NE ČEKAMO nego nastavljamo dalje

Primjer - prikaz opsega temperature


```
        ORG  0
    B   GLAVNI

    ORG  0x1C          ; Adresa prekidnog potprograma za FIQ

    ; ne spremamo kontekst jer koristimo registre od načina FIQ
FIQ   LDR  R8, RTC       ; R8 je bazna adresa RTC-a
    LDR  R9, GPIO1       ; R9 je bazna adresa GPIO-a

    STR  R10, [R8, #8]   ; prihvat prekida RTC-a
    MOV  R10, #0         ; reinicijalizacija RTC-a
    STR  R10, [R8, #0xC]

 ; provjeri spremnost temperaturnog uredaja na bitu 6
    LDR  R11, [R9, #0]
    TST  R11, #0b01000000
    SUBEQS PC, LR, #4    ;ako nema nove temp povratak iz obrade prekida
```

Primjer - prikaz opsega temperature


```
    AND  R12, R11, #0b00111111   ; izdvoji samo bitove 0-5
IMPULS ; kratki impuls na bitu 7
    ORR  R11, R11, #0b10000000   ; digni bit 7 u jedan
    STR  R11, [R9, #0]
    BIC  R11, R11, #0b10000000   ; vrati bit 7 u nulu
    STR  R11, [R9, #0]

LED   LDR  R11, GPIO2
    CMP  R12,#23
    MOVHI R12,#0x20
    STRHI R12, [R11,#0]
    SUBHIS PC, LR, #4      ; ako temp>23 povratak iz obrade prekida
    CMP  R12,#20
    MOVLO R12,#0x80
    STRLO R12, [R11,#0]
    SUBLOS PC, LR, #4      ; ako temp<20 povratak iz obrade prekida
    MOV R12,#0x40
    STR R12, [R11,#0]
    SUBS PC, LR, #4      ; povratak iz obrade prekida
```

Primjer - prikaz opsega temperature


```
GLAVNI ; inicijaliziraj stogove
     MSR  CPSR, #0b11010001  ; prelazak u način rada FIQ
     MOV  R13,  #0x10000     ; inicijalizacija R13_fiq

     MSR  CPSR, #0b11010011  ; prelazak u način rada SVC
     MOV  R13,  #0xFC00      ; inicijalizacija R13_svc

     ; inicijalizacija RTC-a
RTC_I  LDR  R1, RTC          ; R1 je bazna adresa RTC-a

     MOV  R0, #20            ; konstanta brojenja
     STR  R0, [R1, #4]       ; upis u MR

     MOV  R0, #1             ; dozvoli RTC-u da zahtijeva prekid
     STR  R0, [R1, #0x10]    ; upis u CR

     MOV  R0, #0             ; briši brojilo u RTC-u
     STR  R0, [R1, #0xC]     ; upis u LR
```

Primjer - prikaz opsega temperature


```
GPIO_I ; inicijalizacija GPIO

      LDR R1, GPIO1         ; R1 = GPIO bazna adresa
      MOV R0, #0b10000000    ; smjer vrata A, bit 7 je
      STR R0, [R1, #8]      ; izlazni, ostali su ulazni

      LDR  R1, GPIO2        ; R1 je bazna adresa GPIO2
      MOV  R0, #0xE0         ; bitovi 5-7 izlazni
      STRB R0, [R1, #8]

; omogućavanje prihvata prekida
      MRS  R0, CPSR          ; omogućiti prekid FIQ
      BIC  R0, R0, #0x40
      MSR  CPSR, R0

PETLJA B    PETLJA          ; „koristan posao”


RTC    DW   0xFFFF0E00      ; bazne adrese RTC-a i GPIO-a
GPIO1  DW   0xFFFF0F00              ; adresa GPIO-a
GPIO2  DW   0xFFFF0B00
```

Posluživanje više prekidnih vanjskih jedinica

Posluživanje više prekidnih vanjskih jedinica

- Do sada smo vidjeli samo najjednostavniji slučaj kad je na procesor spojena jedna prekidna VJ
- Kad postoji više prekidnih VJ, one se mogu posluživati s ili bez gniježđenja (engl. nesting):
  - bez gniježđenja prekidnih potprograma:
    - dok se poslužuje jedna VJ, drugi prekidi se ne prihvaćaju
    - jednostavniji slučaj
  - s gniježđenjem prekidnih potpograma:
    - dok se poslužuje jedna VJ, može se prihvatiti drugi prekid (većeg prioriteta)
    - kompliciraniji slučaj

Prioriteti

- Svim vanjskim jedinicama treba dodijeliti različite prioritete, što se može napraviti:
  - sklopovski
    - sam procesor ima više prekidnih priključaka s različitim prioritetima (FIQ je prioritetniji od IRQ)
    - prioritetni lanac vanjskih jedinica (daisy-chain)
    - jedinica za kontrolu prekida (interrupt controller)
  - programski (moguće za one VJ koji sklopovski imaju isti prioritet)
  - bilo koja kombinacija prethodno navedenih načina

- Prioriteti imaju dvojaku ulogu:
  - kod istovremenih prekida prioritet određuje kojoj VJ će se prihvatiti prekid (i kod gniježđenja i bez gniježđenja prekida)
  - za vrijeme obrade jednog prekida prioritet određuje hoće li se prihvatiti novi prekid (služi samo za gniježđenje prekida)

Određivanje uzročnika prekida

- Bez obzira kako se poslužuju, uvijek treba odrediti uzročnike prekida
  → o tome ovisi koju VJ ćemo poslužiti, tj. koji prekidni potprogram ili odsječak će biti pozvan

- Opet su moguća različita rješenja - ovisno o procesoru, načinu spajanja vanjskih jedinica itd.:
  - Adresa prekidnog potprograma ovisi o prekidnom priključku (IRQ i FIQ imaju različite adrese prekidnih potprograma 0x18 i 0x1C)
  - Programski se ispituje koja jedinica je izazvala prekid i poziva se odgovarajući odsječak/potprorgam
  - Sklopovski se odabire adresa potprograma što je ujedno i određivanje uzročnika prekida
    - VJ utječe na odabir adrese potprograma
    - Jedinica za kontrolu prekida utječe na odabir adrese potprograma

ARM - više prekidnih vanjskih jedinica

- Kod ARM-a imamo dvije razine prioriteta na dva priključka i dva različita prekidna potprograma koji se odabiru automatski
- Osim toga, određivanje uzročnika prekida i prioritete ćemo ostvariti programski*:
  - uzročnik prekida se određuje **ispitivanjem spremnosti** svih prekidnih VJ koje su spojene na **isti priključak**
    - ovo **ne treba miješati s ispitivanjem uvjetnih VJ**, jer se ovdje samo jednom ispita spremnost, tj. nema čekanja da VJ postane spremna niti se obavlja prozivanje
  - prioriteti su implicitno definirani redoslijedom ispitivanja spremnosti VJ:
    - jedinice koje se prije ispituju imaju veći prioritet

* ARM ima i jedinice za upravljanje prekidima, ali njih nećemo razmatrati

ARM - više prekidnih vanjskih jedinica

- Gniježđenje prekidnih potprograma koristit ćemo samo za jedinice koje su spojene na različite prekidne priključke:
  - Jedinice spojene na FIQ moći će prekinuti izvođenje prekidnog potprograma od jedinica koje su spojene na IRQ
  - Jedinice koje su spojene na isti prekidni priključak neće se moći međusobno prekidati*

* To je moguće ostvariti programski (što je relativno komplicirano) ili korištenjem jedinica za upravljanje prekidima

Primjer - dva RTC-a na IRQ

RTC1 i RTC2 su spojeni na IRQ. RTC1 ima viši prioritet od RTC2.

RTC1 broji impulse dok ih ne dođe 100.

RTC2 mjeri vremenske intervale od 1 sekunde (na ulazu je signal od 1 kHz).

Program treba svake sekunde za jedan povećati sadržaj memorijske lokacije SEKUNDE. Svaki puta kad se odbroji 100 impulsa, treba vratiti sadržaj lokacije SEKUNDE natrag u nulu. Ovo se ponavlja beskonačno.

Idejno rješenje:

Kad prekidnom potprogramu za IRQ prvo ispitamo spremnost od RTC1 (jer ima veći prioritet) i poslužimo ga ako je spreman.

Ako RTC1 nije spreman, onda je sigurno prekid došao od RTC2 pa ga možemo poslužiti bez ispitivanja spremnosti.

Primjer - dva RTC-a na IRQ


```
    ORG  0
    B    GLAVNI

    ORG  0x18            ; adresa prekidnog potprograma
    B    PREKID

GLAVNI ; inicijaliziraj stogove
    MSR  CPSR, #0b11010010  ; prelazak u način rada IRQ
    MOV  R13,  #0x10000     ; inicijalizacija R13_irq

    MSR  CPSR, #0b11010011  ; prelazak u način rada SVC
    MOV  R13,  #0xFC00      ; inicijalizacija R13_svc

    BL   UI_INIT            ; inicijalizacija oba RTC-a

    MRS  R0, CPSR           ; omogućiti prekid IRQ
    BIC  R0, R0, #0x80
    MSR  CPSR, R0

PETLJA B    PETLJA
```

Primjer - dva RTC-a na IRQ


```
UI_INIT; inicijalizacija oba RTC-a
    STMFD SP!, {R0-R2}
    LDR   R1, RTC1        ; dohvat baznih adresa RTC-ova
    LDR   R2, RTC2
    MOV   R0, #0
    STR   R0, [R1, #0x0C]   ; LR - brisanje oba brojila
    STR   R0, [R2, #0x0C]
    MOV   R0, #100
    STR   R0, [R1, #0x04]   ; MR - konstanta za RTC1 - 100 impulsa
    MOV   R0, #1000
    STR   R0, [R2, #0x04]   ; MR - konstanta za RTC2 - 1000 impulsa
    MOV   R0, #1
    STR   R0, [R1, #0x10]   ; CR - prekidanje za oba RTC-a
    STR   R0, [R2, #0x10]
VAN   LDMFD SP!, {R0-R2}
    MOV   PC, LR


RTC1   DW   0xFFFF1000       ; proizvoljno odabrane bazne adrese
RTC2   DW   0xFFFF2000       ; (jer nisu zadane)
```

Primjer - dva RTC-a na IRQ


```
SEKUNDE DW  0     ; brojač sekundi


PREKID ; povećavanje ili resetiranje brojača sekundi
       ; (ovisno o izvoru prekida)
       STMFD SP!, {R0-R2}
       LDR   R1, RTC1      ; dohvat baznih adresa RTC-ova
       LDR   R2, RTC2
       ; određivanje izvora prekida
CHK1   LDR   R0, [R1, #8]  ; provjeri spremnost od RTC1
       CMP   R0, #1
       BEQ   SRV1          ; ako je RTC1 spreman, onda ga posluži
       B     SRV2          ; inače posluži RTC2 (sigurno je spreman)
       ; povratak iz prekidnog potprograma
VAN    LDMFD SP!, {R0-R2}
       SUBS  PC, LR, #4
```

(nastavak na sljedećem slajdu)

Primjer - dva RTC-a na IRQ

(nastavak s prethodnog slajda)

```
    ; posluživanje RTC1 s resetiranjem brojača sekundi
SRV1  STR   R0, [R1, #8]     ; dojava prihvata prekida
    MOV   R0, #0
    STR   R0, [R1, #0xC]    ; LR - brisanje brojila

    STR   R0, SEKUNDE       ; resetiranje brojača SEKUNDE

    B     VAN


    ; posluživanje RTC2 s povećanjem brojača sekundi
SRV2  STR   R0, [R2, #8]     ; dojava prihvata prekida
    MOV   R0, #0
    STR   R0, [R2, #0xC]    ; LR - brisanje brojila

    LDR   R0, SEKUNDE       ; povećaj brojač SEKUNDE
    ADD   R0, R0, #1
    STR   R0, SEKUNDE

    B     VAN
```

Primjer - RTC-a na IRQ i RTC na FIQ

Isti zadatak kao prethodni ali su RTC-ovi drugačije spojeni na prekide:

RTC1 je spojen na FIQ. RTC2 je spojen na IRQ.

RTC1 broji impulse dok ih ne dođe 100.

RTC2 mjeri vremenske intervale od 1 sekunde (na ulazu je signal od 1 kHz).

Program treba svake sekunde za jedan povećati sadržaj memorijske lokacije SEKUNDE. Svaki puta kad se odbroji 100 impulsa, treba vratiti sadržaj loacije SEKUNDE natrag u nulu. Ovo se ponavlja beskonačno.

Idejno rješenje:

- RTC1 automatski ima veći prioritet jer je spojen na FIQ.
- Ovdje uzročnika prekida ne treba tražiti jer je na svaki priključak spojen samo jedan izvor prekida.

Primjer - RTC-a na IRQ i RTC na FIQ


```
    ORG  0
    B    GLAVNI

    ORG  0x18          ; adresa prekidnog potprograma
    B    POVECAJ

    ORG  0x1C
    ; posluživanje RTC1 s resetiranjem brojača sekundi
    LDR   R8, RTC1         ; dohvat bazne adrese od RTC1
SRV1  STR   R9, [R8, #8]     ; dojava prihvata prekida
    MOV   R9, #0
    STR   R9, [R8, #0xC]    ; LR - brisanje brojila

    STR   R9, SEKUNDE       ; resetiranje brojača SEKUNDE
    SUBS  PC, LR, #4


UI_INIT; potprogram za inicijalizaciju oba RTC-a

    ... potpuno isto kao u prethodnom zadataku
```

Primjer - RTC-a na IRQ i RTC na FIQ


```
GLAVNI ; inicijaliziraj stogove (u FIQ se ne koristi stog)

    MSR  CPSR, #0b11010010  ; prelazak u način rada IRQ
    MOV  R13, #0x10000      ; inicijalizacija R13_irq

    MSR  CPSR, #0b11010011  ; prelazak u način rada SVC
    MOV  R13, #0xFC00       ; inicijalizacija R13_svc

    BL   UI_INIT            ; inicijalizacija oba RTC-a

    MRS  R0, CPSR           ; omogućiti prekide IRQ i FIQ
    BIC  R0, R0, #0b11000000
    MSR  CPSR, R0

PETLJA B    PETLJA

    ; proizvoljno odabrane bazne adrese (jer nisu zadane)
RTC1   DW   0xFFFF1000
RTC2   DW   0xFFFF2000

SEKUNDE DW  0  ; brojač sekundi
```

Primjer - RTC-a na IRQ i RTC na FIQ


```
POVECAJ; povećavanje brojača sekundi

     STMFD SP!, {R0,R2}

     LDR   R2, RTC2      ; dohvat bazne adrese od RTC2

     ; posluživanje RTC2 s povećanjem brojača sekundi
SRV2   STR   R0, [R2, #8]      ; dojava prihvata prekida
     MOV   R0, #0
     STR   R0, [R2, #0xC]    ; LR - brisanje brojila

     LDR   R0, SEKUNDE       ; povećaj brojač SEKUNDE
     ADD   R0, R0, #1
     STR   R0, SEKUNDE

     ; povratak iz prekidnog potprograma
VAN    LDMFD SP!, {R0,R2}
     SUBS  PC, LR, #4
```

Čekajući prekid

- Kad za vrijeme obrade jednog prekida dođe zahtjev za drugim prekidom koji ne bude prihvaćen zbog nižeg prioriteta, onda taj drugi prekid nazivamo čekajućim prekidom (pending interrupt)
- Tada se prvi prekid obrađuje do kraja, a za to vrijeme stalno postoji zahtjev za drugim prekidom
- Kad se prvi prekid obradi do kraja, izlazi se iz njegovog prekidnog potprograma i vraća se izvođenje u glavni program čime će se opet dozvoliti prihvaćanje prekida
- Tada se prihvaća čekajući prekid i odmah se poziva prekidni potprogram za njegovu obradu

- U predzadnjem zadatku se može javiti čekajući prekid i od RTC1 i od RTC2 ako dođe za vrijeme obrade drugoga prekida (nema gniježđenja)
- U zadnjem zadatku se može javiti čekajući prekid samo od RTC2 jer je RTC1 prioritetniji