DMA prijenos

Izravni pristup memoriji

Sklopovski UI prijenos

DMA prijenos

- Izravni pristup memoriji ili **DMA** (Direct Memory Access) je **sklopovski** ulazno-izlazni prijenos

- Prijenos ne obavlja procesor odnosno program, nego poseban DMA-sklop (DMA-jedinica, DMA-kontroler / *DMA controller*)

- Glavna karakteristika: relativno velika brzina prijenosa

- DMA-sklop uvijek prenosi podatak između **izvora** (*source*) i **odredišta** (*destination*). Postoje sljedeće kombinacije (mogućnosti prijenosa ovise o konkretnom DMA-sklopu):
  - memorija ⟶ vanjska jedinica
  - memorija ⟶ memorija
  - vanjska jedinica ⟶ memorija
  - vanjska jedinica ⟶ vanjska jedinica

DMA prijenos – osnovna ideja

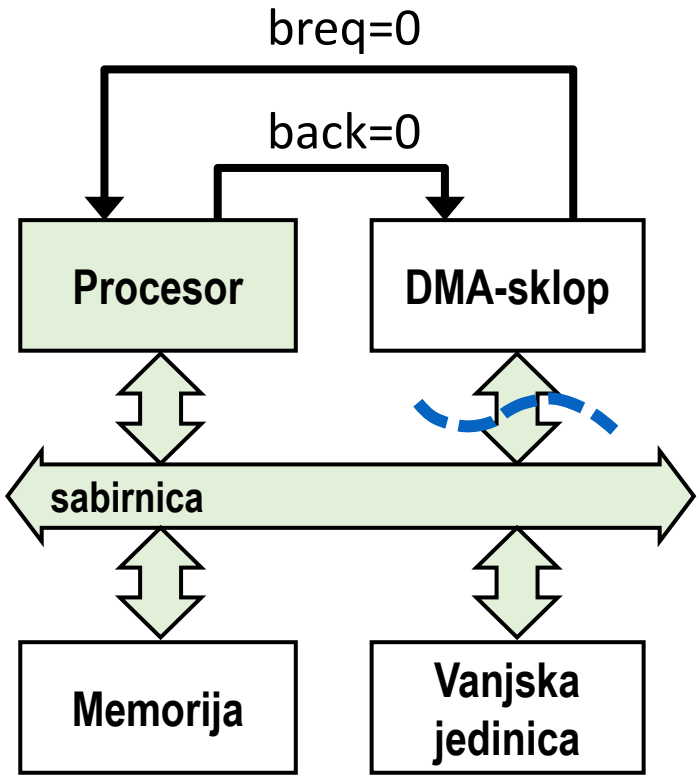
- Procesor prvo inicijalizira DMA-sklop
  - time sklop zna koliko podataka treba prenijeti, gdje su izvor i odredište podataka itd.

- DMA-sklop preuzme upravljanje nad sabirnicom i obavlja prijenos
  - procesor je za vrijeme prijenosa neaktivan

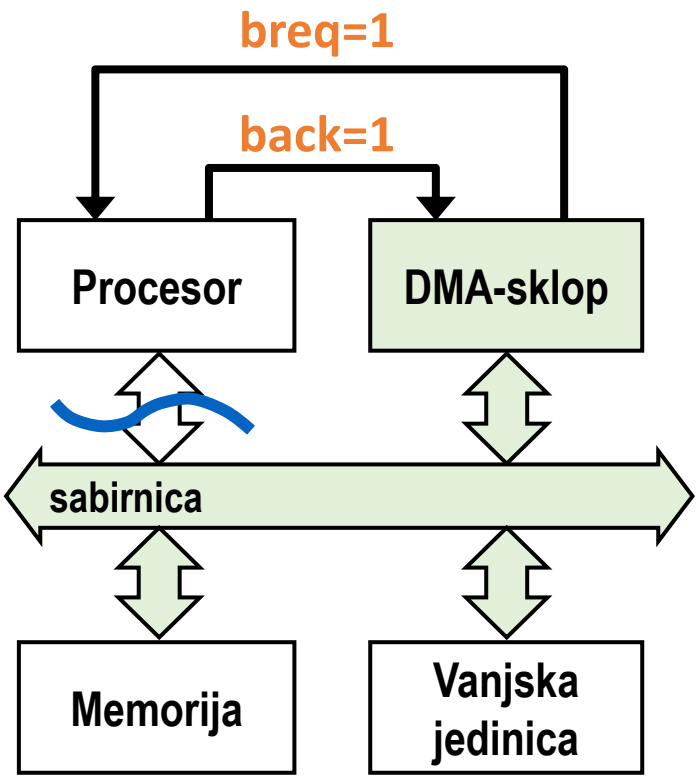
- Kad je DMA-prijenos gotov, DMA-sklop o tome obavještava procesor (npr. prekidom ili uvjetno)

- Procesor i DMA-sklop moraju se međusobno sinkronizirati kako bi u svakom trenutku samo jedan od njih upravljao sabirnicom
  - najjednostavnija sinkronizacija je korištenje dvije linije koje se obično nazivaju BREQ (*Bus REQuest*) i BACK (*Bus ACKnowledge*)
  - pod upravljanjem sabirnicom misli se na iniciranje sabirničkih transakcija čitanja i pisanja (npr. upravljanje adresnom sabirnicom, linijama *read*, *write*, itd.)

DMA - blok-shema



Procesor upravlja sabirnicom*



DMA-sklop upravlja sabirnicom

*Slika simbolično prikazuje stanje na sabirnici. DMA-sklop ne upravlja sabirnicom, ali mu procesor može normalno pristupati kao i drugim vanjskim jedinicama i memoriji

Upravljanje prijenosom

- DMA-sklop određuje **KADA** će pročitati podatak iz izvora ili upisati podatak u odredište, ali mora biti siguran da su izvor i odredište spremni za prijenos
  - Memorija je uvijek spremna za prijenos
  - Vanjska jedinica nije uvijek spremna za prijenos

- Mogućnosti za osiguravanje spremnosti VJ za prijenos:
  1. procesor (uvjetno ili prekidom) prepozna spremnost VJ i to onda „javi" DMA-sklopu, koji dalje upravlja prijenosom. Efikasno za veći broj podataka.
  2. VJ posebnim linijama izravno zatraži prijenos od DMA-sklopa. Ovisno o složenosti DMA-sklopa i VJ, VJ može zahtijevati prijenos određenog broja podataka, a može čak i dojaviti da nema više podataka za prijenos (tj. da je prijenos gotov).

Brzina DMA-prijenosa

- Približna i idealizirana usporedba bezuvjetnog prijenosa i DMA-prijenosa za N podataka (npr. iz memorije u vanjsku jedinicu):

```
PETLJA LDR   R0, [R2], #4   ; R2 = adresa memorije     3 ciklusa
       STR   R0, [R4]       ; R4 = adresa VJ           2 ciklusa
       SUBS  R3, R3, #1     ; R3 = brojač podataka     1 ciklus
       BNE   PETLJA                                    3 ciklusa
```

- Za N podataka programu treba približno N*9 ciklusa

- DMA-sklopu će trebati N*2 ciklusa: DMA čita podatak preko sabirnice (prvi period) i onda ga zapisuje (drugi period).
- DMA-prijenos je brži cca 4,5 puta

Brzina DMA-prijenosa

- Ako bi vanjska jedinica radila prekidno, onda bi za programski prijenos trebalo uračunati i cijelu obradu prekida pa bi razlika u brzini bila znatno veća

- Kod mnogih stvarnih vanjskih jedinica često je DMA-sklop integriran u VJ pa je za jedan podatak potreban samo jedan ciklus za upis/čitanje iz memorije
  - Na primjer, VJ ima veliki interni međuspremnik (*buffer*) koji se brzo može čitati ili pisati da bi se obavio DMA-prijenos, a VJ obrađuje podatke u međuspremniku neovisno o prijenosu (disk, video memorija, itd.)

- Mi ćemo u nastavku predavanja pokazati slučaj zasebnog DMA-sklopa, a ne DMA integriranog u VJ

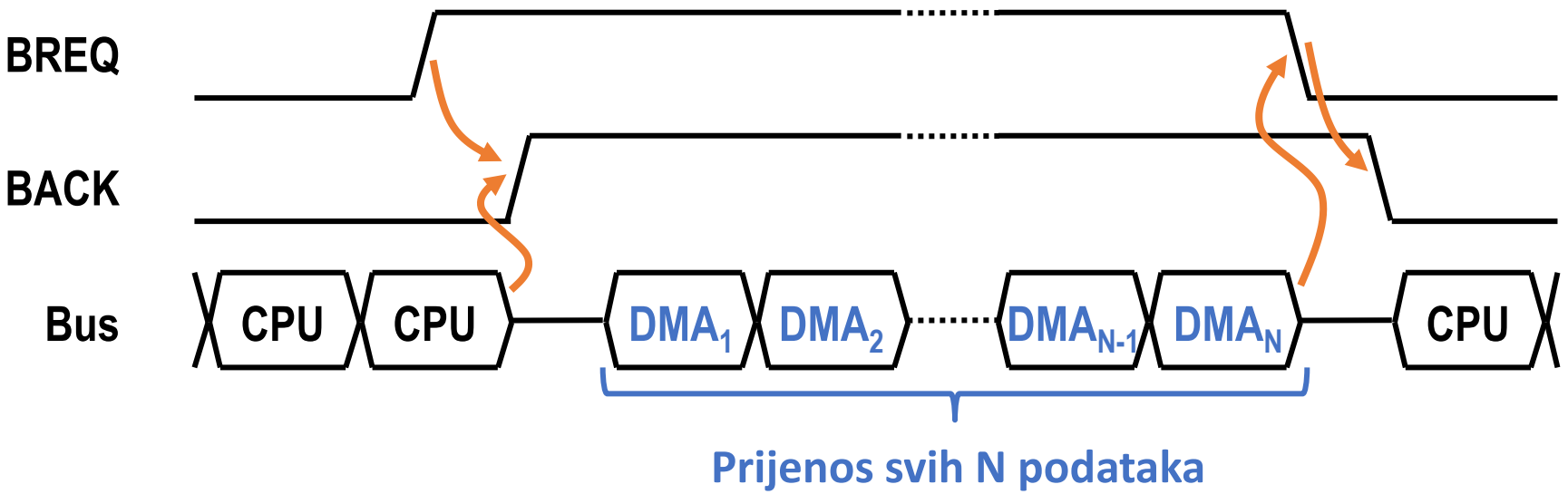
DMA - vrste prijenosa

- Vrste DMA prijenosa:
  - Zaustavljanje procesora (*continuous, halting*)
  - Krađa ciklusa (*cycle stealing, single transfer*)
  - Blokovski (*burst, burst transfer*)
- Konkretni DMA-sklopovi mogu podržavati samo neke od ovih prijenosa

DMA - Zaustavljanje procesora

- DMA **zaustavljanjem procesora** odvija se ovako:
  1. DMA-sklop preuzme upravljanje nad sabirnicom
  2. DMA-sklop prenese **sve podatke**
  3. Tek tada upravljanje nad sabirnicom se vraća procesoru koji je cijelo vrijeme bio zaustavljen
- Tijekom DMA-prijenosa, procesor je potpuno neaktivan jer ne može dohvaćati naredbe iz memorije
  - jedino čeka dojavu od DMA da može nastaviti s radom
- Prednost: najbrži prijenos
- Veliki nedostatak: procesor može dulje vrijeme biti neaktivan
  - što ako dođe neki važni prekid, itd.
- Primjenjivo:
  1. kada broj podataka nije prevelik pa se procesor ne zaustavlja u predugom razdoblju
  2. kada procesor nema važnih poslova

DMA - Zaustavljanje procesora



1. DMA-sklop aktiviranjem BREQ procesoru postavlja „zahtjev za sabirnicom” (BREQ je prioritetniji od zahtjeva za prekid)
2. Na kraju naredbe/ciklusa procesor oslobađa sabirnicu i dojavljuje to DMA-sklopu aktiviranjem BACK
3. DMA preuzima sabirnicu i obavlja DMA-prijenos, a procesor ne radi ništa i samo čeka na deaktiviranje BREQ
4. DMA oslobađa sabirnicu i deaktivira BREQ
5. Procesor ponovno preuzima sabirnicu i deaktivira BACK te nastavlja s radom

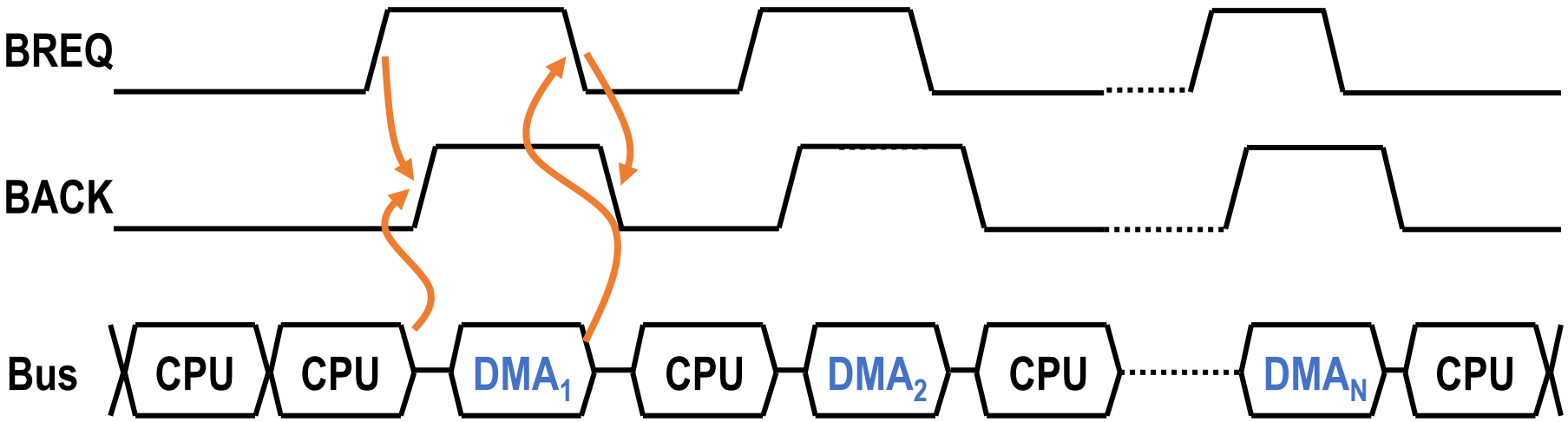
DMA - Krađa ciklusa

- DMA **krađom ciklusa** odvija se ovako:
  1. DMA-sklop preuzme upravljanje nad sabirnicom
  2. DMA-sklop prenese **jedan podatak**
  3. Upravljanje nad sabirnicom se vraća procesoru
  4. Gornja tri koraka se ponavljaju dok se ne prenesu svi podatci

- Prednost: procesor se usporava, ali ipak izvodi glavni program

- Nedostatak: ukupno sporije od zaustavljanja procesora (dio vremena može se trošiti na BREQ/BACK sinkronizaciju)

DMA - Krađa ciklusa



Napomena: broj CPU-ciklusa između dva DMA-ciklusa ovisi o DMA-sklopu

DMA - Blokovski prijenos

- DMA **blokovskim prijenosom** odvija se ovako:
  1. DMA-sklop preuzme upravljanje nad sabirnicom
  2. DMA-sklop prenese **nekoliko podataka (tj. blok)**
  3. Upravljanje nad sabirnicom se vraća procesoru
  4. Gornja tri koraka se ponavljaju dok se ne prenesu svi podatci

- Kompromis između zaustavljanja procesora i krađe ciklusa
  - Zaustavljanje procesora može se promatrati kao blokovski prijenos kod kojeg je veličina bloka jednaka svim podatcima
  - Krađa ciklusa može se promatrati kao blokovski prijenos kod kojeg je u bloku samo jedan podatak
  - Odabirom odgovarajuće veličine bloka može se dobiti „optimalan” prijenos

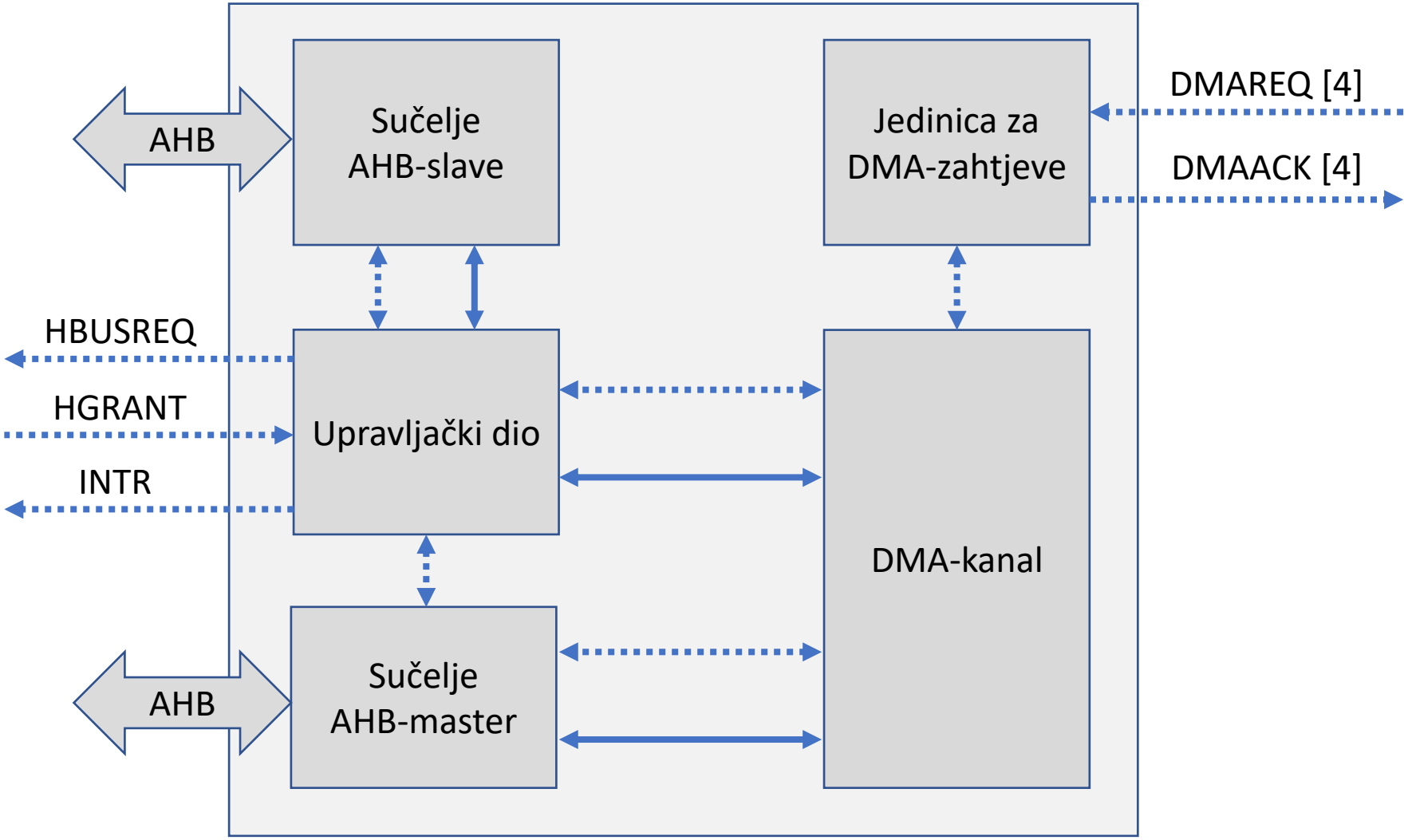
DMAC (DMA Controller)

Osnovno o DMAC-u

- **DMAC** je hipotetski DMA-sklop
  - znatno pojednostavljen stvarni ARM-ov sklop SMDMAC PL 081 (najjednostavniji DMA-sklop za ARM)
- Namjena DMAC-a je brzi prijenos podataka na AHB sabirnici

- Glavna svojstva DMAC-a:
  - Podržava tri tipa transfera: MEM→MEM, MEM→VJ, VJ→MEM
  - Ima samo jedan kanal koji obavlja DMA-prijenos
  - Podržava 4 ulaza za DMA-zahtjeve od strane 4 različite VJ
  - Podržava blokovski prijenos (*burst*) s programabilnom veličinom bloka (ako se odabere 1, efektivno se dobije krađa ciklusa)
  - Ima dva sučelja prema AHB sabirnici (*master* i *slave*)
  - Kada je prijenos gotov:
    - DMAC postaje spreman (tj.postavlja se bistabil stanja)
    - Može postaviti zahtjev za prekid
  - Može prenositi podatak širine bajta, poluriječi ili riječi

Građa DMAC-a


AHB
Sučelje
AHB-slave
Jedinica za
DMA-zahtjeve
DMAREQ [4]
DMAACK [4]
HBUSREQ
HGRANT
INTR
Upravljački dio
DMA-kanal
AHB
Sučelje
AHB-master

Građa DMAC-a

- Dijelovi DMAC-a su:
  - Upravljački dio
    (upravljanje, upravljački registri, generiranje prekida)
  - DMA-kanal (channel)
    (upravljački registri kanala, logika za obavljanje DMA-prijenosa, interni registri i brojači)
  - AHB sučelje
    - Slave-sučelje se koristi kada ARM konfigurira DMAC
    - Master-sučelje se koristi kada se obavlja DMA-prijenos
  - Jedinica za DMA-zahtjeve i odgovore prima zahtjeve za DMA-prijenosom od vanjskih jedinica i šalje im odgovore

DMAC - način rada

1. Prvo treba konfigurirati DMAC, pri čemu se zadaje način rada kanala, adrese izvora i odredišta, broj podataka B u jednom bloku kao i ukupan broj podataka T u transferu
2. Zatim treba omogućiti kanal posebnom upravljačkom riječi pri čemu se interni brojač prenesenih podataka inicijalizira na T
3. VJ postavlja DMA-zahtjev DMAC-u kada je spremna za primanje/slanje B podataka
   a) Ako DMAC prenosi iz memorije u memoriju, onda ovog koraka nema. Nakon 2. koraka DMAC će automatski preskočiti 3. korak
4. DMAC postavlja BUS-zahtjev ARM-u i nakon što mu ARM odobri zahtjev, DMAC preuzme sabirnicu, prenosi jedan blok te vraća sabirnicu ARM-u
5. Interni brojač prenesenih podataka se smanjuje za svaki preneseni podatak
6. Koraci 3-5 se ponavljaju T/B puta (dok interni brojač ne padne na 0)
7. Nakon prijenosa zadnjeg bloka/podatka, onemogućava se kanal, postavlja se stanje spremnosti (i prekid ako je omogućen)

Kanal - konfiguriranje

- Konfiguriranje kanala uključuje (u bilo kojem redoslijedu):
  1. Zadavanje adrese izvora podataka
  2. Zadavanje adrese odredišta podataka
  3. Zadavanje veličine bloka B i veličine transfera T*
  4. Zadavanje konfiguracijske riječi**

- Nakon konfiguriranja kanala, treba omogućiti kanal i tek nakon toga on započinje s radom

---

* B i T izražavaju se u broju „podataka”, neovisno o veličini podatka i pri tome T mora biti djeljiv sa B

** U *konfiguracijskoj riječi* se zadaje veličina podatka kao bajt ili poluriječ ili riječ te niz drugih informacija

Prekidi u DMAC-u

- DMAC postavlja zahtjev za prekidom pomoću priključka INTR
- Prekid se postavlja kada DMAC postane spreman i ako je prekid omogućen u konfiguracijskom registru kanala
- Spremnost znači da je jedan cijeli DMA-transfer završen
- Stanje spremnosti se ogleda u internom status-bistabilu

- Programske mogućnosti za rad s prekidima/stanjem DMAC-a:
  - Može se ispitati stanje spremnosti/prekida
  - Može se obrisati stanje spremnosti (istovremeno i prekida)

Jedinica za DMA-zahtjeve

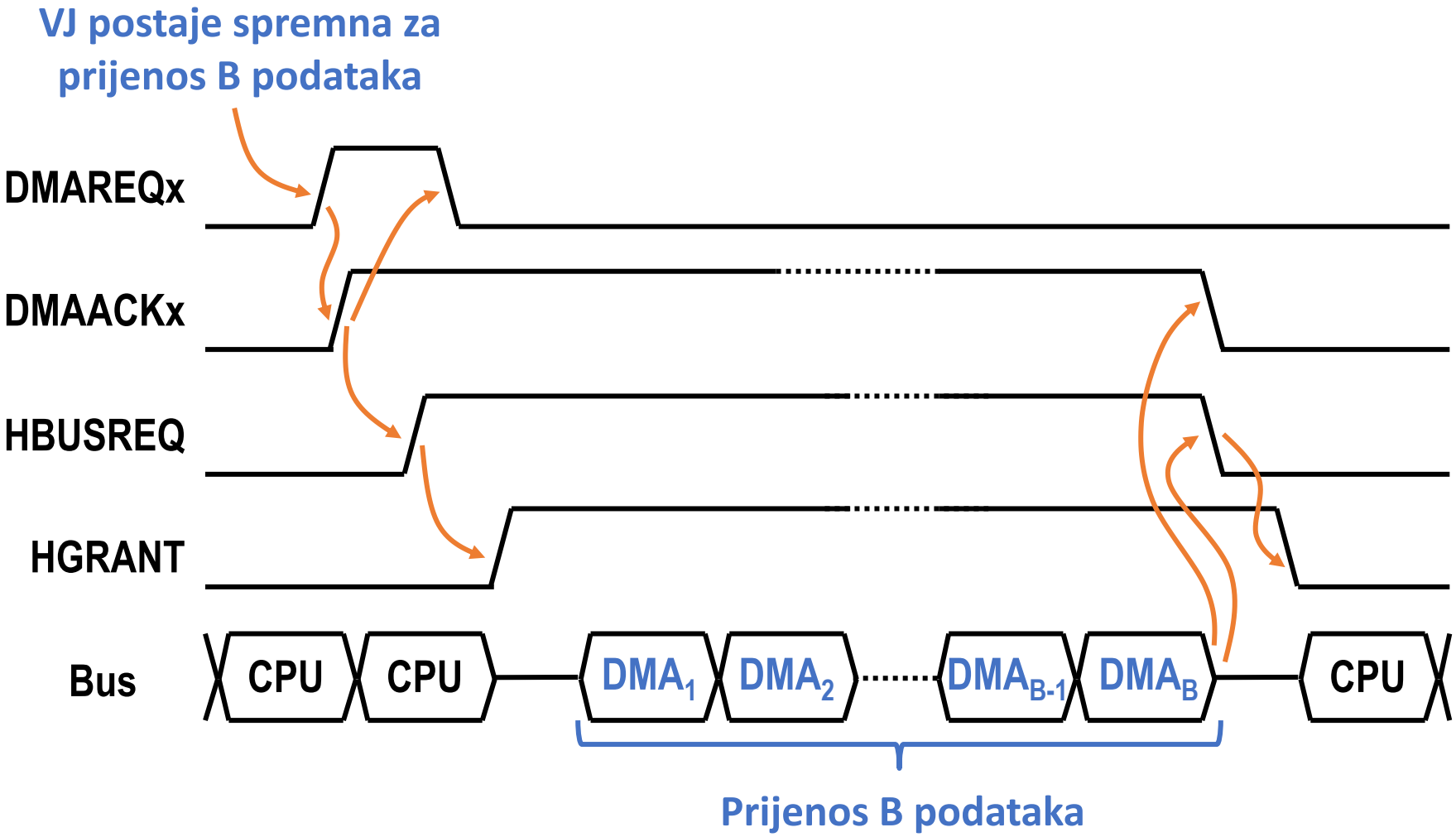
- VJ postavlja DMA-zahtjev DMAC-u kada je spremna za primanje/slanje B podataka (odgovornost programera je da zada ispravan B i T)
- DMA-zahtjev se postavlja preko jednog* od četiriju ulaza DMAREQx (oznaka x znači indeks od 0 do 3)
- Ako je kanal onemogućen ili aktivan (već obavlja prijenos), onda se zahtjev ignorira
- Ako je kanal prenio prethodni blok i čeka na zahtjev za prijenosom sljedećeg bloka, onda će prihvatiti zahtjev i odgovoriti na njega aktiviranjem izlaza DMAACKx.
  Nakon toga započinje prijenos sljedećeg bloka.

---

* U konfiguracijskoj riječi kanala zadaje se preko kojeg od četiriju ulaza DMAREQx će se primati DMA-zahtjevi

Jedinica za DMA-zahtjeve

- Pojednostavljeni način postavljanja DMA-zahtjeva:



Vanjske jedinice pogodne za DMA-prijenos

- Da bi DMA-prijenos bio efikasan, povoljno je:
  - Vanjska jedinica bi trebala biti spremna za prijenos barem nekoliko podataka (u praksi: FIFO s nizom podataka)
  - Vanjska jedinica bi trebala raditi s 32-bitnim podatcima

- Na labosima ćemo koristiti hipotetske vanjske jedinice kako bismo mogli testirati rad DMAC-a
- One će imati FIFO od osam 32-bitnih podataka, a DMA-zahtjeve će davati:
  - za ulaznu jedinicu: kad ima 4 ili više podataka u FIFO-u
  - za izlaznu jedinicu: kad ima 4 ili više slobodnih mjesta u FIFO-u

Adrese DMAC-a

Adresa	Naziv registra	Opis
bazna_adr + 0	Control	1-bitni registar za omogućavanje kanala
bazna_adr + 4	Status/Clear	1-bitni registar stanja spremnosti (ako se čita) 0-bitni registar za brisanje stanja spremnosti (ako se piše)
bazna_adresa + 0x10	SrcAddr	32-bitni registar za adresiranje izvora podataka
bazna_adresa + 0x14	DestAddr	32-bitni registar za adresiranje odredišta podataka
bazna_adresa + 0x18	Sizes	15-bitni registar za zadavanje veličine bloka B i veličine prijenosa T
bazna_adresa + 0x1C	Config	8-bitni registar konfiguracijske riječi kanala

Upravljački registar i registri stanja

Registri kanala

Adrese DMAC-a

- Upravljački registar i registri stanja :
  - **Control**
    - Početna vrijednost registra Control je 0
    - Read (1 bit): čitanje trenutačne omogućenosti kanala (1=omogućen)
    - Write (1 bit): omogućavanje (1) ili onemogućavanje (0) kanala
    - Nakon dovršetka DMA-transfera, bit se automatski briše.
    - Programsko onemogućavanje kanala je moguće (ne koristimo)
      - oprez: trenutačni blok će biti prenesen, a ostali se zanemaruju
  - **Status/Clear**
    - Read (1 bit): čitanje spremnosti, tj. status-bistabila
    - Write (0 bita): brisanje spremnosti ako se upiše bilo koja vrijednost (ujedno brisanje prekida ako je bio postavljen)

Adrese DMAC-a

Registri kanala:

- Sva 4 kanalna registra (SrcAddr, DestAddr, Sizes, Config) su početno 0 i ne smiju se programski mijenjati tijekom DMA-prijenosa

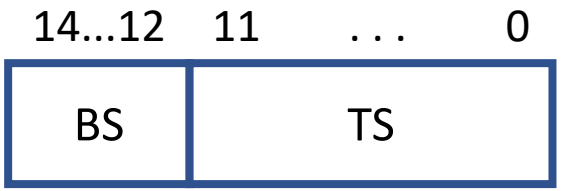
- Iz sva 4 kanalna registra može se čitati njihova trenutačna vrijednost (čitanje ne koristimo u praksi)
  - SrcAddr i DestAddr mijenjaju vrijednost tijekom DMA-prijenosa
  - Sizes i Config ne mijenjaju vrijednost tijekom DMA-prijenosa

Adrese DMAC-a

- **SrcAddr** (32 bita)
  - Zadavanje izvorišne adrese za DMA-prijenos
  - Adresa mora biti poravnata na zadanu širinu podatka 8/16/32
- **DestAddr** (32 bita)
  - Zadavanje odredišne adrese za DMA-prijenos
  - Adresa mora biti poravnata na zadanu širinu podatka 8/16/32

Adrese DMAC-a

- **Sizes**
  - Read/Write (15 bita): čitanje/zadavanje veličine bloka i veličine prijenosa
  - Registar Sizes ima dva polja:
    - 12-bitno polje TS
    - 3-bitno polje BS



- Pomoću TS izravno se zadaje veličina prijenosa T. Ako se zada TS=0, onda je T=4096 (tj. može se prenijeti od 1 do 4096 podataka)

- Pomoću BS zadaje se veličina bloka B po formuli: $B = 2^{BS}$
- Pomoću BS od 0 do 7 zadaju se veličine bloka: 1, 2, 4, 8, 16, 32, 64, 128

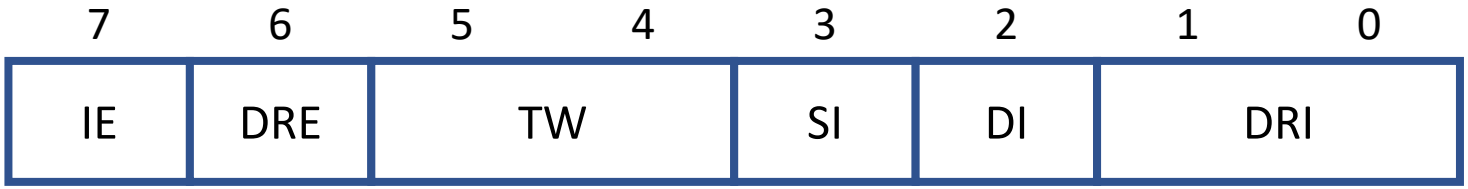
- Ograničenje: T mora biti djeljiv sa B. (tj. i zadnji blok mora biti „pun”)
- Za vrijeme DMA prijenosa se vrijednost registra Sizes ne mijenja
  - Postoji zasebni interni brojač prijenosa koji se inicijalizira na T i odbrojava na dolje.

Adrese DMAC-a

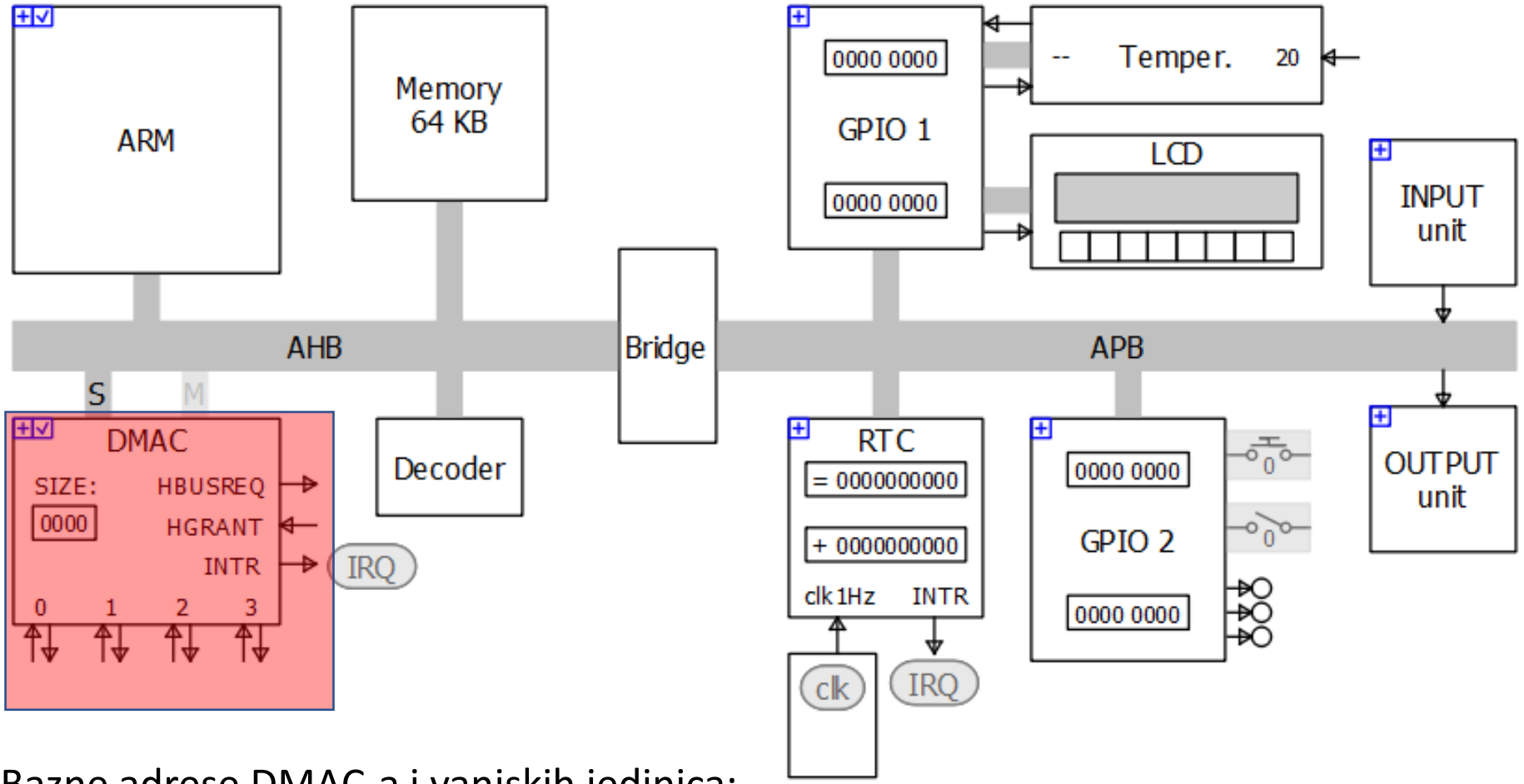
Config

- Read/Write (8 bita): dohvat trenutačne/zadavanje nove konfiguracijske riječi kanala
- Bitovi registra Config su:

- IE [7] - Interrupt Enable. Omogućuje (1) ili onemogućuje (0) postavljanje prekida INTR kad DMAC postane spreman, tj. kad se DMA-transfer završi.
- DRE [6] - DMA-Request Index Enable. Omogućuje (1) ili onemogućuje (0) uporabu priključaka DMA-request (zadanih sa DRI). Za prijenos M->M treba zadati DRE=0.
- TW [5..4] - Transfer Width. Zadaje širinu podatka: 00=bajt, 01=poluriječ, 10=riječ, 11=zabranjena kombinacija. Napomena: veličina bloka i transfera ne zadaju se niti u bajtovima niti u riječima nego u „podatcima" čija se širina definira bitovima TW.
- SI [3] i DI [2] - Source/Destination Increment. Zadaje se hoće li se registar SrcAddr/DestAddr automatski povećavati nakon transfera svakog pojedinog podatka (1) ili će ostati nepromijenjen (0). Napomena: iznos povećanja ovisi o bitovima TW i može biti 1, 2 ili 4.
- DRI [1..0] - DMA-Request Index. Zadaje indeks X preko kojeg dolaze DMA-zahtjevi na priključcima DMAREQx. Ako je DRE=0, ovi bitovi se ne koriste.



DMAC u SSPARCSS-u



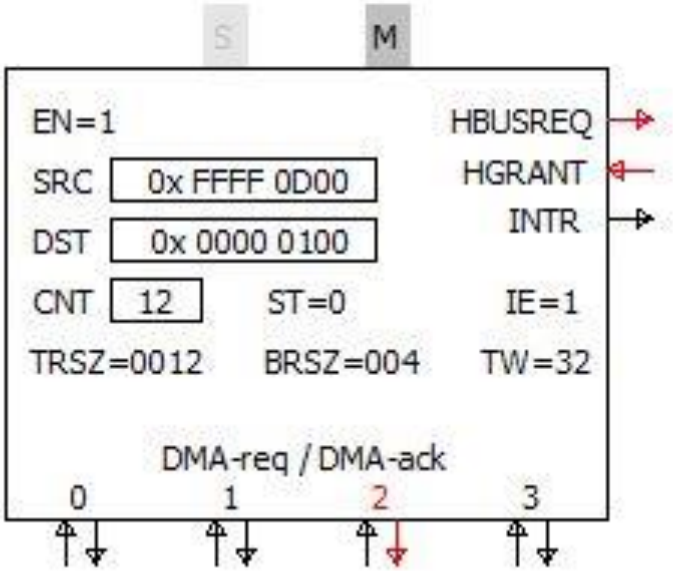
Bazne adrese DMAC-a i vanjskih jedinica:

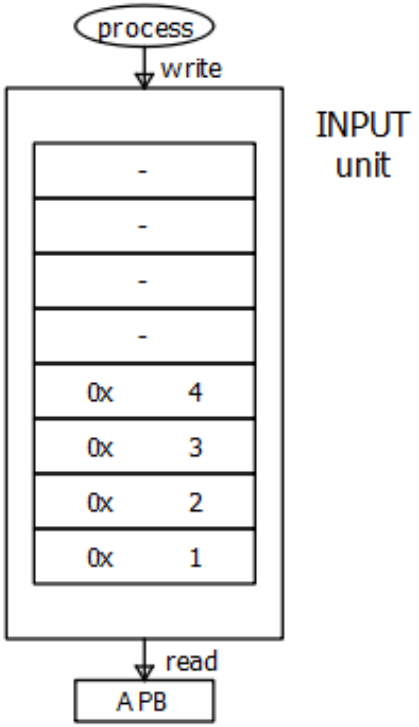
DMAC = 00FF 0000

INPUT unit = FFFF 0D00 (spojena na DMAREQ2/DMAACK2)

OUTPUT unit = FFFF 0C00 (spojena na DMAREQ3/DMAACK3)

DMAC u SSPARCSS-u





Transfer:

SRC[0x FFFF 0D00] ---> data=0x 0000 0001 ---> DEST[0x 0000 0100]

Izgled internog prikaza DMAC-a i ulazne jedinice za vrijeme prijenosa prvog podatka (0x1)

Rječnik za DMAC

- **kanal** - dio DMAC-a koji je zadužen za prijenos podataka iz izvora u odredište. Kanal se konfigurira kako bi mu se zadalo kako treba obaviti prijenos. (napredniji DMAC može imati više kanala koji rade paralelno i imaju međusobne prioritete)
- **izvor i odredište (source, destination)** - komponente iz kojih se čitaju/u koje se pišu podatci. Izvor i odredište mogu biti memorija ili vanjska jedinica, ali u DMAC-u ne mogu oboje biti vanjske jedinice
- **podatak** - podatak koji se prenosi u jednom čitanju i pisanju, može imati različite širine od 1, 2 ili 4 bajta (širina se zadaje u inicijaliziranju DMAC-a i tijekom jednog transfera se ne mijenja)
- **prijenos podatka** - prijenos jednog podatka iz izvora u odredište
- **Transfer ili DMA-transfer** - postupak prijenosa cijelog niza podataka koji se sastoji od određenog broja prijenosa blokova. Nakon transfera kanal postaje neaktivan.
- **prijenos bloka (burst)** - prijenos bloka podataka koji se obavlja nakon jednog DMA-zahtjeva (blok može sadržati samo jedan podatak što onda postaje prijenos jednog podatka)
- **veličina transfera** - ukupni broj podataka T u transferu (zadaje se u inicijaliziranju DMAC-a)
- **veličina bloka ili bursta** - broj podataka B u jednom bloku (zadaje se u inicijaliziranju DMAC-a)
- **DMA-zahtjev (DMA request)** - zahtjev od strane VJ poslan DMAC-u preko priključka DMAREQx. Zahtjev znači spremnost za prijenos jednog bloka podataka. Ako se DMA-zahtjevi ne koriste (npr. i izvor i odredište su memorija), onda DMAC automatski prenosi podatke bez čekanja.
- **zahtjev za sabirnicom (bus request)** - zahtjev od strane DMAC-a poslan ARM-u preko priključka BUSREQ. DMAC će nakon preuzimanja sabirnice prenijeti jedan blok podataka.

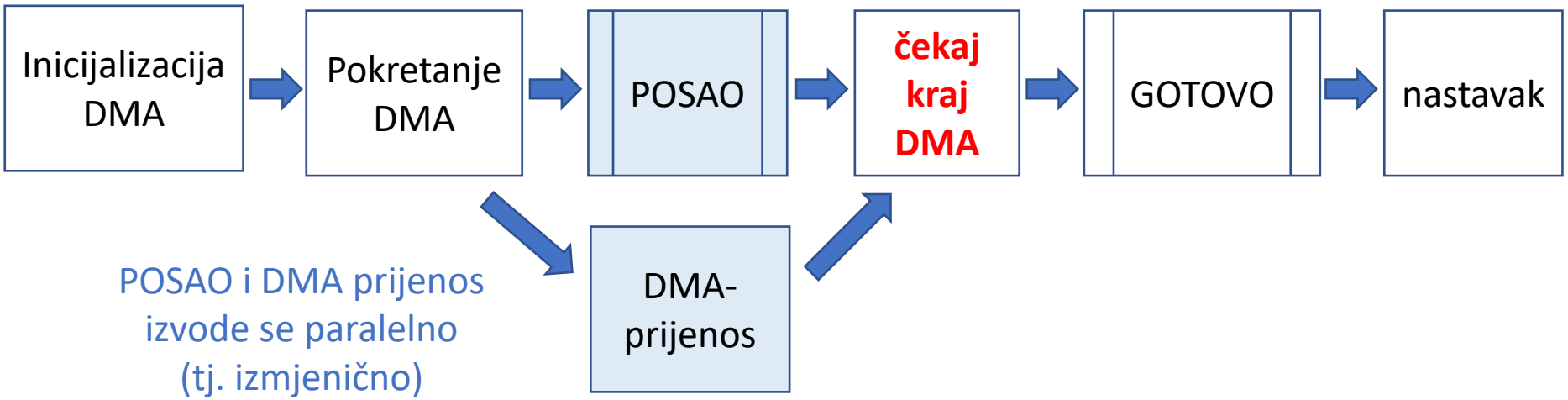
Primjer

DMAC je na adresi 0x00FF0000. DMA-prijenosom treba kopirati memorijski blok od 256 bajtova s adrese 0x1000 na 0x2000. Tijekom prijenosa treba izvoditi potprogram POSAO, a nakon POSAO i nakon dovršenog prijenosa treba pozvati potprogram GOTOVO i zatim nastaviti s glavnim programom. Kraj prijenosa treba ispitati uvjetno. Potprograme POSAO i GOTOVO ne treba pisati.

Rješenje: u zadatku nije zadana širina podatka ni veličina bloka pa ih odabiremo ovako:

- podatak je riječ
- veličina transfera T bit će 64 ***podatka*** (64*4B(riječ) = 256B)
- a blok zadamo da je veličine npr osam podataka

Program će se izvoditi ovako:



Primjer (M->M, uvjetno)


```
      ORG  0
GLAVNI MOV  R13, #0x10000        ; inicijalizacija R13_svc
      LDR  R1, DMAC_A            ; inicijalizacija DMAC-a
      MOV  R0, #0x1000
      STR  R0, [R1,#0x10]        ; Source address
      MOV  R0, #0x2000
      STR  R0, [R1,#0x14]        ; Destination address
      MOV  R0, #0x3040           ; Sizes: burst size = 8 = 2^3
      STR  R0, [R1,#0x18]        ;        transfer size = 0x040
      MOV  R0, #0b00101100
      STR  R0, [R1,#0x1C]        ; Configuration
```

(nastavak na sljedećem slajdu)

IE	DRE	TW	SI	DI	DRI
disable	disable	32 bita	++	++	not used
0	0	10	1	1	00

Primjer (M->M, uvjetno)

(nastavak s prethodnog slajda)

```
      ; omogući kanal, tj. pokretanje DMA
      MOV  R0, #1
      STR  R0, [R1,#0]      ; Control
      BL   POSAO            ; paralelno s DMA-prijenosom
CEKAJ  LDR  R0, [R1, #0x4]    ; dohvat registra Status
      CMP  R0, #0           ; ispitaj najniži bit (0=>nije spreman)
      BEQ  CEKAJ            ; ako DMA-prijenos još traje => čekaj
END    STR  R0, [R1, #0x4]    ; brisanje stanja spremnosti
      BL   GOTOVO           ; DMA-prijenos je gotov => pozovi GOTOVO
NASTAVI  SWI 123456         ; nastavak programa (kraj simulacije)

DMAC_A DW   0x00FF0000      ; adresa DMAC-a


POSAO MOV PC,LR

GOTOVO MOV PC,LR


 ORG 0x1000
      DB 0,1,2,3,……
 ORG 0x2000
      DS 256
```

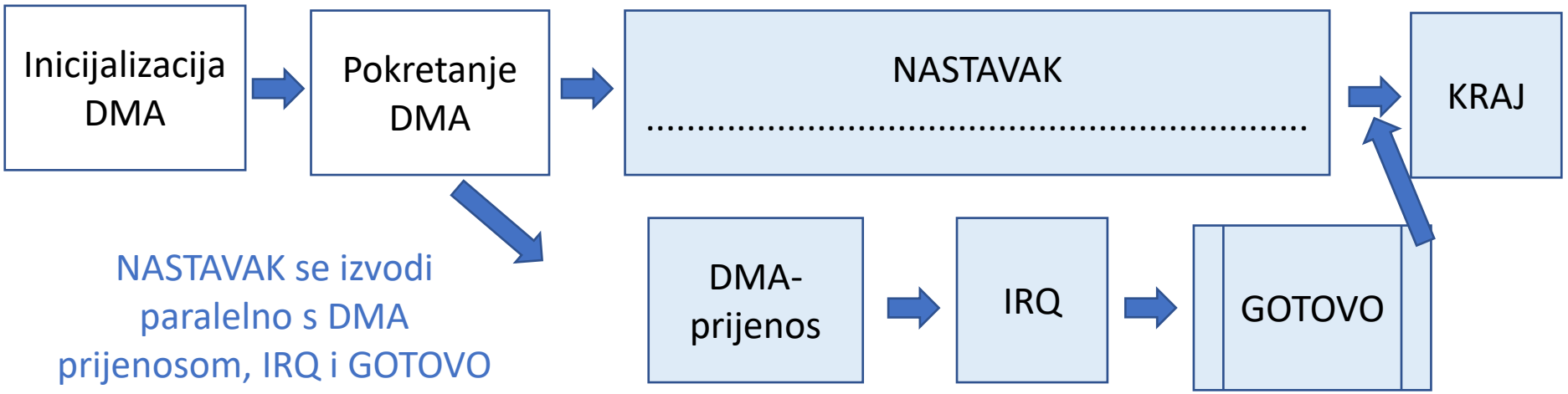
Komentar: programsko uvjetno čekanje na spremnost DMAC-a ovdje nije bilo problem u pogledu efikasnosti, jer je ionako trebalo nakon POSAO pozvati potprogram GOTOVO i nije bilo drugih zadaća koje između toga treba izvoditi.

Primjer (M->M, IRQ)

DMA-prijenosom s krađom ciklusa treba kopirati memorijski blok od 256 bajtova s adrese 0x1000 na 0x2000. Nakon prijenosa treba izvesti potprogram GOTOVO. Neovisno o DMA-prijenosu i potprogramu GOTOVO treba izvoditi ostatak glavnog programa od labele NASTAVAK. Kraj prijenosa treba ustanoviti prekidom IRQ. Nakon obavljenog prijenosa prekinuti izvođenje NASTAVAK.

Rješenje: DMA-prijenos odvija se kao u prošlom zadatku (uz veći broj zahtjeva BUSREQ), ali je redoslijed izvođenja drugačiji

Prenosit ćemo 32-bitne podatke, njih 64, tj. 0x40. Veličina bloka B bit će 1.



Primjer (M->M, IRQ)


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

    ORG   0x18
    B     DMA_KRAJ      ; skok na obradu iznimke IRQ


MAIN   ; Inicijalizacija stogova

    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
```

(nastavak na sljedećem slajdu)

Primjer (M->M, IRQ)

(nastavak s prethodnog slajda)

```
INIT   ; inicijalizacija DMAC-a
     MOV  R1, #0x00FF0000       ; adresa ima oblik rotiranog bajta

     MOV  R0, #0x1000
     STR  R0, [R1,#0x10]        ; Source address

     MOV  R0, #0x2000
     STR  R0, [R1,#0x14]        ; Destination address

     MOV  R0, #0x0040            ; Sizes: burst size = 1 = 2^0
     STR  R0, [R1,#0x18]        ;        transfer size = 0x040

     MOV  R0, #0b10101100
     STR  R0, [R1,#0x1C]        ; Configuration
```

(nastavak na sljedećem slajdu)

RAZLIKA: IE=1

IE	DRE	TW	SI	DI	DRI
enable	disable	32 bita	++	++	not used
1	0	10	1	1	00

Primjer (M->M, IRQ)

(nastavak s prethodnog slajda)

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

      ; omogući kanal, tj. pokretanje DMA
      MOV  R0, #1
      STR  R0, [R1,#0]      ; Control

NASTAVAK                                ; neki glavni program….
       CMP R7, #1               ; provjera da li je DMA gotov
       BNE NASTAVAK
       SWI 123456
```

Primjer (M->M, IRQ)

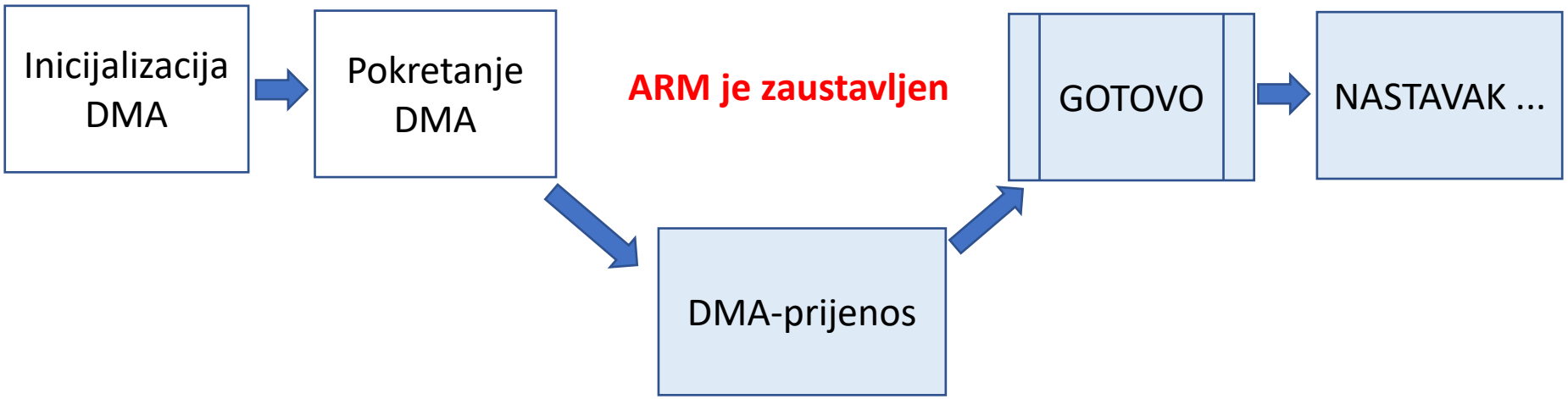

```
        ; posluživanje prekida IRQ

DMA_KRAJ  STMFD SP!, {R1, LR}       ; spremi kontekst

        MOV   R1, #0x00FF0000      ; brisanje statusa, tj. dojava
        STR   R1, [R1, #0x4]       ; prihvata prekida DMAC-u

        BL    GOTOVO               ; nakon DMA-prijenosa => GOTOVO

        LDMFD SP!, {R1, LR}        ; obnovi kontekst
        SUBS  PC, LR, #4           ; povratak iz iznimke/prekida

GOTOVO    MOV R7, #1
        MOV PC,LR

DMAC_A    DW   0x00FF0000       ; adresa DMAC-a
```

Primjer (M->M, zaustavljanje)

DMAC-om treba kopirati memorijski blok od 512 bajtova s adrese 0x1000 na 0x2000. Zadatak riješiti zaustavljanjem procesora.

Nakon prijenosa treba izvesti potprogram GOTOVO. Neovisno o DMA-prijenosu i potprogramu GOTOVO treba izvoditi ostatak glavnog programa od labele NASTAVAK.

Rješenje: Zaustavljanje procesora je moguće jer se veličina od 512 bajtova može zadati registrom Sizes kao B = 128 i T = 128 uz veličinu podatka od 4 bajta. Detekcija kraja ovdje nije potrebna jer se procesor zaustavlja.



Primjer (M->M, zaustavljanje)


```
      ORG  0
GLAVNI MOV  R13, #0x10000        ; inicijalizacija R13_svc
      MOV  R1, #0x00FF0000       ; inicijalizacija DMAC-a
      MOV  R0, #0x1000
      STR  R0, [R1,#0x10]        ; Source address
      MOV  R0, #0x2000
      STR  R0, [R1,#0x14]        ; Destination address
      ; Sizes = 0x7080
      MOV  R0, #0x7000           ; B = 128 = 2^7
      ORR  R0, R0, #0x080        ; T = 128 = 0x080
      STR  R0, [R1,#0x18]
      MOV  R0, #0b00101100
      STR  R0, [R1,#0x1C]        ; Configuration
```

(nastavak na sljedećem slajdu)

IE	DRE	TW	SI	DI	DRI
disable	disable	32 bita	++	++	not used
0	0	10	1	1	00

Primjer (M->M, zaustavljanje)

(nastavak s prethodnog slajda)

```
    ; omogući kanal, tj. pokretanje DMA
    MOV  R0, #1
    STR  R0, [R1,#0]      ; Control


    ; procesor će ovdje biti zaustavljen do dovršetka DMA-prijenosa

    STR  R0, [R1, #0x4]   ; brisanje stanja spremnosti
    BL   GOTOVO           ; nakon DMA-prijenosa

NASTAVAK                  ; nastavak programa

    SWI 123456

GOTOVO MOV PC,LR
```

Komentar: odmah nakon omogućavanja kanala, doći će do pokretanja DMA prijenosa zato jer se prijenos obavlja iz memorije u memoriju. Da se koristila VJ, onda bi DMA bio pokrenut tek kad VJ postavi DMA-zahtijev.

Ulazne i izlazne jedinice za DMAC

- U primjerima i na labosima koristit ćemo hipotetske ulazne i izlazne jedinice prilagođene komunikaciji s DMAC-om
- Svojstva jedinica:
  - Imat ćemo ulazne i izlazne jedinice, obje s analognim ponašanjem
  - Jedinica ima FIFO s 8 riječi (32-bitnih podataka)
  - Jedinica ima dva priključka (izlazni REQ i ulazni ACK) kompatibilna s DMAC-ovim priključcima DMAREQx i DMAACKx
  - Jedinica daje zahtjev za DMA-prijenosom:
    - ulazna jedinica: kada ima 4 ili više podataka u FIFO-u
    - izlazna jedinica: kada ima 4 ili više slobodnih mjesta u FIFO-u
  - Zbog jednostavnijeg i prilagodljivijeg konfiguriranja na labosima, jedinici se zadaje ukupni broj podataka koje će primiti/poslati
  - Vanjski uređaj spojen na jedinicu će određenom brzinom slati/primati podatke i na taj način neovisno o računalu puniti/prazniti FIFO (onoliko podataka koliko smo zadali)

Ulazne i izlazne jedinice za DMAC - adrese

Adresa	Naziv registra	Opis
bazna_adr + 0	Data	32-bitni registar za prijenos podatka: - read-only za Input unit - write-only za Output unit
bazna_adr + 4	TransferSize	32-bitni registar/brojač (write only) - zadavanje broja podataka za prijenos

Ulazne i izlazne jedinice za DMAC

- Adrese ulaznih i izlaznih jedinica:
  - Bazna adresa+0: Data
    - Adresa preko koje računalo (DMAC) čita ili piše 32-bitne podatke podatke sa ulazne jedinice/na izlaznu jedinicu
    - Za ulaznu jedinicu moguće je samo čitanje (read-only)
    - Za izlaznu jedinicu moguće je samo pisanje (write-only)
  - Bazna adresa+4: TransferSize (write-only)
    - Write (32 bita): zadavanje broja podataka koje će vanjski uređaj poslati/primiti od jedinice o čemu ovisi broj zahtjeva za DMA-prijenosom. Nakon zadavanja, jedinica počinje s radom. Ovaj broj se interno smanjuje do nule i tada jedinica opet postaje neaktivna.
    - zadani broj mora biti djeljiv sa 4 zbog jednostavnosti rada FIFO-a i DMA-zahtjeva

Primjer (M->VJ, IRQ)

DMA-prijenosom treba poslati sadržaj memorijskog bloka od 64 riječi s adrese 0x1000 na upravo opisanu hipotetsku izlaznu jedinicu na adresi 0xFFFF0C00. Nakon prijenosa treba izvesti potprogram GOTOVO. Neovisno o DMA-prijenosu i potprogramu GOTOVO treba izvoditi ostatak glavnog programa od labele NASTAVAK. Kraj prijenosa treba ustanoviti prekidom IRQ. DMAC je na adresi 0x00FF0000. Jedinica je spojena na DMAREQ3 i DMAACK3.

Rješenje:

DMAC moramo programirati tako da mu veličina bloka i širina riječi odgovara vanjskoj jedinici, jer inače prijenos neće raditi ispravno. Budući da ulazna jedinica daje DMA-zahtjev kad ima 4 (ili više) podataka spremnih za prijenos, onda obavezno veličinu bloka zadajemo kao 4.

U bit Destination increment obavezno upisujemo 0, jer se vanjskoj jedinici uvijek pristupa na istoj adresi.

Pomoću DRI obavezno zadajemo 3 kao indeks priključaka za DMA-zahtjev jer je izlazna jedinica spojena na te priključke. Također pomoću bita DRE moramo omogućiti korištenje DMA-zahtjeva.

Primjer (M->VJ, IRQ)


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

        ORG   0x18
        B     DMA_KRAJ      ; skok na obradu iznimke IRQ


VJ_ADR  DW    0xFFFF0C00    ; adresa vanjske jedinice

MAIN    ; Inicijalizacija stogova
        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
```

(nastavak na sljedećem slajdu)

Primjer (M->VJ, IRQ)

(nastavak s prethodnog slajda)

```
INIT   MOV  R1, #0x00FF0000      ; inicijalizacija DMAC-a
       MOV  R0, #0x1000
       STR  R0, [R1,#0x10]       ; Source address
       LDR  R0, VJ_ADR
       STR  R0, [R1,#0x14]       ; Destination address
       MOV  R0, #0x2040           ; Sizes: B = 4 = 2^2
       STR  R0, [R1,#0x18]       ;       T = 64 = 0x040
       MOV  R0, #0b11101011
       STR  R0, [R1,#0x1C]       ; Configuration
```

(nastavak na sljedećem slajdu)

IE	DRE	TW	SI	DI	DRI
enable	enable	32 bita	++	ne	3

Primjer (M->VJ, IRQ)

(nastavak s prethodnog slajda)

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

    ; inicijaliziraj jedinicu (pošalji joj broj podataka 64)
    LDR  R2, VJ_ADR
    MOV  R0, #64
    STR  R0, [R2, #4]

    ; omogući kanal, tj. pokretanje DMA
    MOV  R0, #1
    STR  R0, [R1,#0]      ; Control

NASTAVAK
     CMP R7, #1      ; nastavak glavnog programa koji
     BNE NASTAVAK    ; se izvodi paralelno s DMA-prijenosom
     SWI 123456
```

(nastavak na sljedećem slajdu)

Primjer (M->VJ, IRQ)


```
        ; posluživanje prekida IRQ
DMA_KRAJ  STMFD SP!, {R1, LR}    ; spremi kontekst
        MOV   R1, #0x00FF0000  ; brisanje statusa, tj. dojava
        STR   R1, [R1, #0x4]   ; prihvata prekida DMAC-u
        BL    GOTOVO           ; nakon DMA-prijenosa izvedi GOTOVO
        LDMFD SP!, {R1, LR}    ; obnovi kontekst
        SUBS  PC, LR, #4       ; povratak iz iznimke/prekida
```

Komentar: iako bi se prema veličini transfera moglo zaključiti da je ovdje moguće koristiti zaustavljanje procesora, to ipak nije slučaj. Razlog je što ova vanjska jedinica radi „po blokovima” od 4 riječi i nema na raspolaganju odjednom slobodnog mjesta za primanje svih 64 riječi.

Komentar: kao i u prethodnom primjeru, i ovdje bi programsko uvjetno čekanje na spremnost DMAC-a bilo neefikasno, jer nakon pokretanja prijenosa bi se gubilo vrijeme na čekanje dovršetka prijenosa, umjesto da se odmah započne izvoditi nastavak glavnog programa od labele NASTAVAK

Primjer za vježbu (M->disk, IRQ)

Disk sa bufferom/međuspremnikom je spojen na DMAC. Buffer se vidi kao 32-bitna memorija kapaciteta 1024 riječi na adresi FFFF4000. Disk nije spojen na DMA-zahtjeve, ali se spremnost diska može očitati s adrese FFFF5000 (spremnost znači da je buffer „prazan”). Spremnost diska se mora obrisati programski (slanjem bilo kojeg podatka) i to nakon „punjenja buffera” (tj. nakon DMA-prijenosa).

Treba DMA-prijenosom poslati na disk sadržaj bloka memorije s adrese 0x1000 i duljine 1024 riječi. Nakon prijenosa treba obrisati blok 0x1000. Zadatak riješiti blokovskim prijenosom (po 16 riječi) i prekidom IRQ. Glavni program cijelo vrijeme treba pozivati potprogram POSAO (ne treba ga pisati)

Rješenje:

Kod konfiguriranja DMAC-a ćemo pomoću DI za odredište podataka zadati da ima automatsko povećavanje adrese (jer se disku pristupa kao memoriji - na uzastopnim lokacijama). Budući da disk ne postavlja zahtjeve za DMA, moramo ih onemogućiti pomoću DRE (bitovi DRI se ne koriste).

Prije pokretanja DMA-prijenosa, u glavnom programu čekat ćemo spremnost diska. Kad DMA bude gotov, onda ćemo obrisati spremnost diska.

Primjer (M->disk, IRQ)


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

    ORG   0x18
    B     DONE          ; skok na obradu iznimke IRQ

DSK_B  DW    0xFFFF4000      ; adresa buffera od diska
DSK_S  DW    0xFFFF5000      ; adresa statusa od diska


MAIN   ; Inicijalizacija stogova

    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
```

(nastavak na sljedećem slajdu)

Primjer (M->disk, IRQ)

(nastavak s prethodnog slajda)

```
INIT   MOV  R1, #0x00FF0000      ; inicijalizacija DMAC-a
       MOV  R0, #0x1000
       STR  R0, [R1,#0x10]       ; Source address
       LDR  R0, DSK_B
       STR  R0, [R1,#0x14]       ; Destination address
       MOV  R0, #0x4400           ; Sizes: B = 16 = 2^4
       STR  R0, [R1,#0x18]       ;       T = 1024 = 0x400
       MOV  R0, #0b 1 0 10 1 1 00
       STR  R0, [R1,#0x1C]       ; Configuration
```

(nastavak na sljedećem slajdu)

IE	DRE	TW	SI	DI	DRI
enable	disable	32 bita	++	++	not used
1	0	10	1	1	00

Primjer (M->disk, IRQ)

(nastavak s prethodnog slajda)

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

    ; čekaj da disk javi spremnost
    LDR  R2, DSK_S
CEKAJ  LDR  R0, [R2]      ; čekaj spremnost diska - uvjetno
    CMP  R0, #0
    BEQ  CEKAJ


    ; omogući kanal, tj. pokretanje DMA
    MOV  R0, #1
    STR  R0, [R1,#0]    ; Control

LOOP   BL   POSAO         ; nastavak glavnog programa koji se izvodi
    B    LOOP          ; paralelno s DMA-prijenosom i obradom prekida
```

Primjer (M->disk, IRQ)


```
        ; posluživanje prekida IRQ
DONE  STMFD SP!, {R1-R3}       ; spremi kontekst
    MOV   R1, #0x00FF0000     ; brisanje statusa, tj. dojava
    STR   R1, [R1, #0x4]      ; prihvata prekida DMAC-u
    LDR  R1, DSK_S            ; brisanje spremnosti diska jer
    STR  R1, [R1]             ; je DMA-prijenos gotov
    MOV   R1, #0x1000         ; brisanje bloka memorije
    MOV   R2, #1024
    MOV   R3, #0
BRISI STR   R3, [R1], #4
    SUBS  R2, R2, #1
    BNE   BRISI
    LDMFD SP!, {R1-R3}        ; obnovi kontekst
    SUBS  PC, LR, #4          ; povratak iz iznimke/prekida
```

Komentar: ovdje bi programsko uvjetno čekanje na spremnost DMAC-a bilo neefikasno, jer nakon pokretanja prijenosa bi se gubilo vrijeme na čekanje dovršetka prijenosa, umjesto da se odmah započne pozivati potprogram POSAO.