ARH1 glavne fore

FRISC

Stog

```
GLAVNI MOVE 10000, SP
```

Potprogrami

```
IME PUSH R1 ; kontekst
 ...
 POP R1 ; kontekst
 RET
```

Prekidi

Za razliku od običnih potprograma, kod prekidnog potprograma kontekst uključuje i registar stanja SR (ako se mijenja). SR se jedino može MOVE-ati.

```
 ...
 MOVE SR, R0 ; spremanje i SR zbog AL naredaba
 PUSH R0
 ...
 POP R0 ; obnavljanje SR-a
 MOVE R0, SR
 ...
```

INT

Adresa za prekidni potprogram je na **ORG 8**. U glavnom programu je potrebno dozvoliti maskirajući prekid postavljanjem zastavice GIE u registru SR.

```
MOVE %B 10000, SR ; dozvoljavanje prihvata prekida na INT
```

NMI

Prekidni potprogram je na **ORG 0C**. Nije potrebno dozvoliti NMI, jer je on inicijalno dozvoljen, a programski ga nije ni moguće zabranjivati i dozvoljavati.

Vanjske jedinice općenito

Koristimo adrese **FFFF0000** pa dalje u skladu s dogovorom o adresama rezerviranim za vanjske

jedinice kod procesora FRISC

Bezuvjetne

Samo LOAD i STORE rokaj.

```
BVJ1 EQU 0FFFF1000 ; adrese vanjskih jedinica
BVJ2 EQU 0FFFF2000
 ORG 0
GLAVNI LOAD R0, (BVJ1) ; učitati podatak iz BVJ1
 ...
 STORE R0, (BVJ2) ; spremanje na BVJ2
 HALT
```

Uvjetne

```
UVJ1POD EQU FFFF1000 ; adrese vanjskih jedinica
UVJ1BS EQU FFFF1004 ; bistabil stanja
UVJ2POD EQU FFFF2000
UVJ2BS EQU FFFF2004 ; bistabil stanja

 ORG 0
 ; provjera spremnosti UVJ1
BS1 LOAD R0, (UVJ1BS) ; učitavanje bistabila stanja (BS)
 AND R0, 1, R0 ; ispitati BS od UVJ1
 JR_Z BS1 ; ako je 0, čekati da UVJ1 postane spremna
 LOAD R0, (UVJ1POD) ; učitava se podatak iz UVJ1
 STORE R0, (UVJ1BS) ; slanjem bilo kojeg podatka briše se BS
 ...
 ; provjera spremnosti UVJ2
BS2 LOAD R1, (UVJ2BS) ; učitavanje BS
 AND R1, 1, R1 ; ispitati BS od UVJ2
 JR_Z BS2 ; ako je 0, čekati da UVJ2 postane spremna
 STORE R0, (UVJ2POD) ; spremanje podatka na UVJ2
 STORE R0, (UVJ2BS) ; slanjem bilo kojeg podatka briše se BS
 HALT
```

Napomena: *Treba paziti na jedan detalj, a to je trenutak čitanja podatka iz bezuvjetne vanjske jedinice. Ovaj podatak se čita u zadnjem trenutku, neposredno prije nego će biti upotrijebljen (poslan uvjetnoj jedinici). Na ovaj način, uvjetnoj jedinici bit će poslan „najaktualniji“ podatak.*

Prekidne

LOAD-amo IACK i gledamo je li 1, tako utvrđujemo koja je vanjska jedinica uzrkovala prekid ako ih imamo više.

INT

Adresa za prekidni potprogram je na **ORG 8**. Postoji podatak, IACK, IEND i STOP. U glavnom

programu je potrebno dozvoliti maskirajući prekid postavljanjem zastavice GIE u registru SR.

```
MOVE %B 10000, SR ; dozvoljavanje prihvata prekida na INT
```

```
PVJ1POD EQU 0FFFF0000 ; lokacija za prijenos podataka
PVJ1IACK EQU 0FFFF0004 ; lokacija za dojavu prihvata prekida
PVJ1IEND EQU 0FFFF0008 ; lokacija za dojavu kraja posluživanja
PVJ1STOP EQU 0FFFF000C ; lokacija za zaustavljanje PVJ1
 ; POČETAK IZVOĐENJA
 ORG 0
POCETAK MOVE 10000, SP ; inicijalizacija stoga
 JR GLAVNI ; preskakanje vektora
 ; DEFINIRANJE PREKIDNOG VEKTORA (TJ. ADRESE PREKIDNOG POTPROGRAMA)
 ORG 8 ; vektor za INT mora biti na adresi 8
 DW 200 ; vektor je proizvoljan, ovdje je to 200
```

Na adresi prekidnog vektora počinje program:

```
PREKIDNI PUSH R0 ; spremanje konteksta
 STORE R0, (PVJ1IACK) ; dojava prihvata prekida
 LOAD R0, (PVJ1POD) ; učitavanje podatka
 ...
 STORE R0, (PVJ1IEND) ; dojava kraja obrade prekida
 POP R0 ; obnova konteksta
 RETI ; bitno je koristiti RETI umjesto RET
```

NMI

Prekidni potprogram je na **ORG 0C**. Postoji podatak, IACK, IEND i STOP. Nije potrebno dozvoliti NMI, jer je on inicijalno dozvoljen, a programski ga nije ni moguće zabranjivati i dozvoljavati.

```
PVJ1POD EQU 0FFFF0000 ; lokacija za prijenos podataka
PVJ1IACK EQU 0FFFF0004 ; lokacija za dojavu prihvata prekida
PVJ1IEND EQU 0FFFF0008 ; lokacija za dojavu kraja posluživanja
PVJ1STOP EQU 0FFFF000C ; lokacija za zaustavljanje PVJ1
 ; POČETAK IZVOĐENJA
 ORG 0
POCETAK MOVE 10000, SP ; inicijalizacija stoga
 JP GLAVNI ; preskakanje prekidnog potprograma
 ; NEMASKIRAJUĆI PREKIDNI POTPROGRAM MORA BITI NA ADRESI 0C
 ORG 0C ; prekidni potprogram za NMI (nemaskir. prekid)
PREKIDNI PUSH R0 ; spremanje konteksta
 STORE R0, (PVJ1IACK) ; dojava prihvata prekida
 LOAD R0, (PVJ1POD) ; učitavanje podatka
 ...
 STORE R0, (PVJIEND) ; dojava kraja obrade prekida
 ...
 POP R0 ; obnova konteksta
 RETN ; povratak iz nemaskirajućeg prekida
```

FRISC-GPIO

GPIO može raditi u 4 načina rada od kojih dva omogućavaju sinkroni prijenos podataka a dva asinkroni: - ulazni (ulazni, sinkroni) - ispitivanje bitova (ulazni, asinkroni) - izlazni (izlazni, sinkroni) - postavljanje bitova (izlazni, asinkroni)

31–24	23 – 16	15 – 8	7 – 5	4	3	2	1 – 0
-	ACTIVE	MASK	-	AND/OR	VRSTA INT	INT	MODE
	0 – aktivna je 0 1 – aktivna je 1			0 – OR 1 – AND	0 – maskirajući 1 – nemaskirajući	0 – ne postavlja prekid 1 – postavlja prekid	00 – izlazni način 01 – ulazni način 10 – postavljanje bitova 11 – ispitivanje bitova

```
GPIOC EQU 0FFFF0000 ; adresa upravljačkog registra
GPIOD EQU 0FFFF0004 ; adresa za čitanje/pisanje podataka
GPIOSTAT EQU 0FFFF0008 ; ispitivanje/brisnje spremnosti
GPIOEND EQU 0FFFF000C ; dojava kraja posluživanja
 ORG 0
GLAVNI MOVE 5000, R0 ; adresa za upisivanje u blok memorije
 MOVE 0, R1 ; brojač primljenih podataka
INIT_GPIO MOVE %B 001, R2 ; GPIO inicijalizacija: 0 – bez prekida,
 ; 01 – ulazni način
 STORE R2, (GPIOC)
CEKAJ LOAD R2, (GPIOSTAT) ; čitanje stanja spremnosti
 AND R2, 1, R2 ; ako je spremnost=0, treba čekati dalje
 JR_Z CEKAJ
POSLUZI LOAD R2, (GPIOD) ; učitavanje podataka
 STORE R2, (GPIOSTAT) ; brisanje stanja spremnosti
 ...
 STORE R2, (GPIOEND) ; kraj posluživanja
 HALT
```

FRISC-CT

Maksimalna vrijednost konstante koja se može upisati je **65535**. FRISC-CT broji prema dolje. Nakon što dođe do nula opet broji od iste konstante.

Kao uvjetna:

```
CTCR EQU 0FFFF0000 ; proizvoljno odabrana bazna adresa
CTLR EQU 0FFFF0004
CTSTAT EQU 0FFFF0008
CTEND EQU 0FFFF000C
 ORG 0
GLAVNI MOVE %D 2000, R0 ; vremensku konstantu 2000
 STORE R0, (CTLR) ; ... stavljamo u LR
 MOVE %B 01, R0 ; upravljačka riječ=01, CT ne postavlja prekid
 STORE R0, (CTCR) ; ... i brojilo broji
```

```
PETLJA LOAD R0, (CTSTAT) ; čitanje stanja spremnosti
 AND R0, 1, R0 ; ako je spremnost=0 ...
 JR_Z PETLJA ; ... treba čekati dalje
 ; inače je spremnost=1, tj. vrijeme je prošlo
GOTOVO STORE R0, CTSTAT ; brisanje stanja spremnosti CT
 ...
 STORE R0, CTEND ; dojava kraja posluživanja CT-u
 HALT ; zaustavljanje procesora
```

Kao prekidna i broji impulse:

```
CTCR EQU 0FFFF0000 ; adrese vanjskih jedinica
CTLR EQU 0FFFF0004
CTIACK EQU 0FFFF0008
CTIEND EQU 0FFFF000C
 ORG 0
POCETAK MOVE 10000, SP ; inicijalizacija stoga
 JP GLAVNI ; skok u glavni program
 ORG 8 ; prekidni vektor
 DW 1000
GLAVNI MOVE %D 350, R2 ; konstanta brojenja
 STORE R2, (CTLR)
 MOVE %B 011, R2 ; upravljačka riječ za CT:
 STORE R2, (CTCR) ; INT priključak, CT broji, postavlja se prekid
 MOVE %B 10000, SR ; dozvola prekida u procesoru na priključku INT
PETLJA JP PETLJA ; beskonačna petlja
 ORG 1000 ; adresa prekidnog potprograma
PREKIDNI PUSH R0 ; spremanje konteksta
 MOVE SR, R0
 PUSH R0
 STORE R0, (CTIACK) ; dojava prihvata prekida
 ...
 STOP ; ili zaustavimo ili samo javimo kraj posluzivanja
 MOVE %B 0, R0 ; nova upravljačka riječ:
 STORE R0, (CTCR) ; CT više ne broji
 ...
DALJE STORE R0, (CTIEND) ; dojava kraja prekida
 POP R0 ; obnova konteksta s povratkom
 MOVE R0, SR
 POP R0
 RETI
```

FRISC-DMA

bitovi 31 – 4	bit 3	bit 2	bit 1	bit 0
-	DESTINATION	SOURCE	MODE	INT
	0 – memorija 1 – vanjska jedinica	0 – memorija 1 – vanjska jedinica	0 – zaustavljanje procesora 1 – krađa ciklusa	0 – ne postavlja prekid 1 – postavlja prekid

Vrste DMA prijenosa: - Zaustavljanje procesora (engl. continuous, halting) - Krađa ciklusa (engl. cycle stealing, word-at-a-time) - Blokovski (engl. burst)

Inicijalizacija

```
DMA_SRC EQU 0FFFF0000 ; definiranje adresa za sklop DMA
DMA_DEST EQU 0FFFF0004
DMA_SIZE EQU 0FFFF0008
DMA_CTRL EQU 0FFFF000C
DMA_START EQU 0FFFF0010
DMA_ACK EQU 0FFFF0014

 ORG 0
INIT_DMA MOVE 2000, R0 ; slanje adrese izvorišnog bloka
 STORE R0, (DMA_SRC)
 MOVE 5000, R0 ; slanje adrese odredišnog bloka
 STORE R0, (DMA_DEST)
 MOVE %D 1000, R0 ; slanje veličine bloka
 STORE R0, (DMA_SIZE)
 MOVE %B 0010, R0 ; slanje upravljačke riječi: mem->mem,
 STORE R0, (DMA_CTRL) ; ...krađa ciklusa, bez prekida
 STORE R0, (DMA_START) ; pokretanje DMA-prijenosa
 ; čekanje da sklop DMA prenese blok
CEKAJ LOAD R0, (DMA_ACK) ; čitanje stanja spremnosti
 AND R0, 1, R0 ; ako je spremnost=0, treba čekati dalje
 JR_Z CEKAJ
 STORE R0, (DMA_ACK) ; brisanje spremnosti DMA
 ; brisanje izvorišnog bloka
 ...
 HALT
```

ARM

Općenito

Inicijalni skok na glavni program (naredba **B GLAVNI**) preko prekidnog potprograma potreban je da se po pokretanju računala ne bi odmah izveo prekidni potprogram.

```
ORG 0
B GLAVNI ; skok na glavni program
```

Posebnosti atlasa

Pretpostavljena (default) baza za pisanje brojeva	heksadekadska (osim za definiranje iznosa pomaka i rotacija) MOV R0, #16 – ovdje je 16 u heksadekadskoj bazi MOV R0, #16<8 – ovdje je 16 u heksadekadskoj bazi, a 8 u dekadskoj bazi MOV R0, R0, LSL # 16 – ovdje je 16 u dekadskoj bazi
Pisanje heksadekadskog broja	%H
Pisanje dekadskog broja	%D
Pisanje binarnog broja	%B
Pseudonaredbe za definiranje sadržaja memorije	Bajt: DW, DB Poluriječ: DH Riječ: DW Prostor: DS (napomena: DB, DH i DW poravnavaju se na adresu djeljivu sa 4)
Pisanje neposredne vrijednosti u naredbama za obradu podataka (operand <immed_8r>)	baza(hex)<rotacija_ulijevo(dekadski) npr. 3A<8 je zapis heksadekadskog broja 3A00
Naredba za čitanje/upis podatka sa memorijske lokacije označene labelom	LDR{cond}{B|SB|H|SH} Rd, labela STR{cond}{B|H} Rd, labela (napomena: naredba se prevodi tako da se koristi PC kao bazni registar, a odmak se računa automatski)
Definiranje niza registara u naredbama LDM i STM	U popisu registri moraju biti razdvojeni zarezima i navedeni u rastućem redoslijedu, npr. {R2,R4,R15}. Oblik {Rx-Ry} nije dozvoljen, izuzetak je navođenje {R0-R15}
Zaustavljanje procesora	HALT ili SWI 123456

Uvjeti

Uvjet	Značenje (engl.)	Način ispitivanja zastavica
EQ	*Equal*	Z
NE	*Not equal*	!Z
CS/HS	*Carry Set / Unsigned higher or same*	C
CC/LO	*Carry Clear / Unsigned lower*	!C
MI	*Negative*	N
PL	*Positive or zero*	!N
VS	*Overflow*	V
VC	*No overflow*	!V
HI	*Unsigned higher*	C AND !Z
LS	*Unsigned lower or same*	!C OR Z
GE	*Signed greater than or equal*	N == V
LT	*Signed less than*	N != V
GT	*Signed greater than*	!Z AND N == V
LE	*Signed less than or equal*	Z OR N != V
AL	*Always (normally omitted)*	-

Povratak iz potprograma

```
MOV PC, LR
```

Prekidi

U glavnom programu *načelno treba omogućiti prihvaćanje prekida tek nakon inicijalizacije svih vanjskih jedinica i ostalih parametara*, jer bi se inače moglo dogoditi da neka od njih postavi prekid, a da još nisu inicijalizirane sve vanjske jedinice, brojači i ostali parametri programa.

Primijetite da se, za razliku od FRISC-a, kod ARM-a prihvaćanje pojedinih prekida zadaje upisom „0“ u odgovarajući bit registra CPSR (bit 6 za FIQ, bit 7 za IRQ). (Naredba **BIC**)

Povratak iz potprograma

```
SUBS PC, R14, #4 ; povratak iz prekidnog potporgama
```

FIQ

Ako je zadan zadatak sa brzim prekidom, dobro je koristiti činjenicu da su registri **R8–R14** višeznačno definirani, te da se adresiranjem ovih registara u načinu rada FIQ, zapravo pristupa registrima R8_fiq – R14_fiq, pa ih nije potrebno spremati na stog (jer su ti registri nevidljivi po povratku u glavni program, odnosno u korisnički način rada).

```
ORG 1C ; adresa brzog prekida
; ovdje se direktno pise potprogram
```

```
MRS R0, CPSR
BIC R0, R0, #40 ; dozvoliti prekid FIQ
MSR CPSR_c, R0
```

IRQ

```
ORG 18 ; adresa prekida
B PREKIDNI ; obicno se ovo napravi i ako nema FIQ
```

```
MRS R0, CPSR
BIC R0, R0, #80 ; dozvoliti prekid IRQ
MSR CPSR_c, R0
```

Stog

```
MOV R13, #1<16 ; inicijalizacija stoga
```

Kontekst

```
STMFD R13!, {R0, R14} ; spremanje konteksta
...
LDMFD R13!, {R0, R14} ; povratak konteksta
```

Ukoliko imamo poziv potprograma unutar potprograma **OBAVEZNO** spremiti i **R14**

Vanjske jedinice

Najčešće ćemo u zadatcima koristiti najviše memorijske adrese, slično kao kod FRISC-a (npr. adresu

FFFF0000). Budući da ARM nema apsolutno adresiranje u memorijskim naredbama nego samo registarsko, ovakve se adrese ne mogu izravno zadati u naredbama STR, pa ih valja prvo upisati u neki registar koji će se koristiti kao bazni registar prilikom pristupa dotičnoj vanjskoj jedinici.

GPIO

Adresa	Naziv registra	Opis
GPIO_bazna_adr	GPIOPADR	8-bitni registar podataka, vrata A
GPIO_bazna_adr + 4	GPIOPBDR	8-bitni registar podataka, vrata B
GPIO_bazna_adr + 8	GPIOPADDR	8-bitni registar smjera podataka za vrata A
GPIO_bazna_adr + C	GPIOPBDDR	8-bitni registar smjera podataka za vrata B

A - ulazna vrata, B - izlazna vrata.

Primijetite da se čitanje i pisanje na vrata sklopa GPIO zapravo izvodi bezuvjetno, jer je GPIO bezuvjetna jedinica koja nema bistabil stanja ni mogućnost postavljanja prekida.

Inicijalizacija

Kada postavimo jedinicu na bilo koji od 8 bitova u registar smjera za A ili B onda okrenemo smjer.

```
; inicijalizacija GPIO-a
MOV R3, #0FF ; npr. okrenemo svih 8
STR R3, [R2, #0C] ; smjer GPIO B
STR R3, [R2, #08] ; smjer GPIO A
```

RTC

Adresa	Naziv registra	Opis
RTC_bazna_adr	RTCDR	32-bitni registar podataka (može se samo čitati)
RTC_bazna_adr + 4	RTCMR	32-bitni registar usporedbe
RTC_bazna_adr + 8	RTCSTAT/RTCEOI	1-bitni registar stanja prekida (ako se čita) 0-bitni registar za brisanje prekida (ako se piše)
RTC_bazna_adr + C	RTCLR	32-bitni registar za punjenje brojila
RTC_bazna_adr + 10	RTCCR	1-bitni upravljački registar

RTC, za razliku od FRISC-ovog CT-a, ne ponavlja automatski prethodni ciklus brojenja pa ga je potrebno ponovno inicijalizirati. Brojilo (u registru **RTCLR**) se neće automatski vratiti na ništicu, pa ga treba programski obrisati.

Inicijalizacija

```
LDR R1, [R2], #4 ; R1 = bazna adresa RTC-a
; inicijalizacija sklopa RTC
LDR R3, [R2] ; dohvatiti konstantu iz memorije...
STR R3, [R1, #4] ; ...i spremiti je u RTCMR
MOV R2, #0
STR R2, [R1, #0C] ; obrisati brojilo RTCLR
MOV R2, #1
STR R2, [R1, #10] ; omogućiti RTC da generira prekid
```

Reinicijalizacija

U nekakvom prekidnom programu recimo:

```
; reinicijalizacija RTC-a
STR R0, [R1, #8] ; brisanje RTCINT
MOV R0, #0 ; brojač na 0
STR R0, [R1, #0C] ; spremanje na RTCLR
```

Računanje konstante

- f - frekvencija (u Hz)
- t - vrijeme koje želimo (u sekundama)
- C - konstanta

```
C = t * f
```

Npr. 10 kHz i 2 sekunde. f = 10000, t = 2. Treba brojati do 20000

Npr. 15 MHz i 50ms. f = 15 * 10^6, t = 50 * 10^-3. Treba brojati do 750 000.

Temperaturni sklop sa sikronizacijom

- Bitovi 6 i 7 služe za sinkronizaciju. Bitovi 6 i 7 su aktivni u visokoj razini.
- *Bit 6 ulazi u GPIO* i pomoću njega temperaturni sklop signalizira da je izmjerio temperaturu i postavio njezinu vrijednost na nižih šest bitova. Kad procesor ustanovi da je stanje na bitu 6 u jedinici, može očitati temperaturu s nižih šest bitova.
- Nakon što je pročitao podatke, procesor pomoću *izlaznog bita 7* dojavljuje temperaturnom sklopu da je vrijednost temperature pročitana i da treba izbrisati bit 6.

```
CEKAJ_TEMP LDR R2, [R1, #4] ; čekaj spremnost temperaturnog sklopa
ANDS R2, R2, #%B 01000000
BEQ CEKAJ_TEMP
CITAJ_TEMP LDR R2, [R1, #4] ; čitaj temperaturu (nižih 6 bitova)
AND R2, R2, #%B 00111111
STRB R2, [R4], #1 ; spremi temperaturu u memoriju
ORR R2, R2, #%B 10000000 ; postavi bit 7 u jedan
STR R2, [R1, #4]
```

```
AND R2, R2, #%B 01111111 ; postavi bit 7 u nulu
STR R2, [R1, #4]
```

LCD

Znak poslan LCD-u se zapravo zapisuje u njegov interni registar na desnu poziciju, a svi prethodni znakovi se pomiču ulijevo (dotadašnji znak s lijeve pozicije se gubi).

U potprogramu LCDWR njemu se ostvaruje potrebna sinkronizacija s LCD-om i to pomoću najvišeg bita. Da bi se znak poslao LCD-u, potrebno je prvo ASCII kôd znaka postaviti na najnižih 7 bitova, a zatim na najvišem bitu generirati pozitivni impuls (promijeniti bit iz 0 u 1 i zatim natrag iz 1 u 0). Ovaj impuls signalizira LCD-u da treba zapamtiti nižih 7 bita u svojem internom registru.

- na LCD je poslan znak **0D**, koji će obrisati interni registar LCD-a.
- treba poslati cijeli tekst naposljetku ga prikazati (paziti da se znak za prikaz teksta **0A** pošalje tek nakon što je svih 8 znakova teksta pohranjeno u interni registar LCD-a).