

# ARM Šalabahter

Skracenic	
{cond}	Pogledaj Tablicu <b>Polje uvjeta {cond}</b>
<Opnd2>	Pogledaj Tablicu <b>Operand 2</b>
<fields>	Pogledaj Tablicu <b>PSR polja</b>
{S}	Ako postoji, naredba osvježava zastavice
C*, V*	Zastavica je nepredvidiva nakon izvođenja naredbe kod arhitekture ARM v4 i ranijih arhitektura
<immed_8r>	32-bitna konstanta, dobije se rotacijom u desno 8-bitne vrijednosti za paran broj bitova
<immed_8*4>	10-bitna konstanta, dobije se lijevim posmakom 8-bitne vrijednosti za dva bita

Načini adresiranja	
<a_mode2>	Pogledaj Tablicu <b>Načini adresiranja 2</b>
<a_mode2P>	Pogledaj Tablicu <b>Načini adresiranja 2 (Post-indeks)</b>
<a_mode3>	Pogledaj Tablicu <b>Načini adresiranja 3</b>
<a_mode4L>	Pogledaj Tablicu <b>Načini adresiranja 4 (Block load or Stack pop)</b>
<a_mode4S>	Pogledaj Tablicu <b>Načini adresiranja 4 (Block store or Stack push)</b>
<a_mode5>	Pogledaj Tablicu <b>Načini adresiranja 5</b>
<reglist>	Lista registrara, odvojeni zarezima a unutar vitičastih zagrada
{!}	Ako je napisan !, tada se bazni registar osvježava nakon prijenosa podatka
\$	<b>ARM arhitektura</b>

Polje uvjeta {cond}	
Oznaka	Opis
EQ	Equal
NE	Not equal
CS / HS	Carry Set / Unsigned higher or same
CC / LO	Carry Clear / Unsigned lower
MI	Negative
PL	Positive or zero
VS	Overflow
VC	No overflow
HI	Unsigned higher
LS	Unsigned lower or same
GE	Signed greater than or equal
LT	Signed less than
GT	Signed greater than
LE	Signed less than or equal
AL	Always (normally omitted)

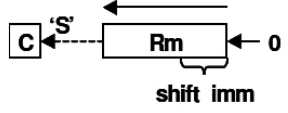
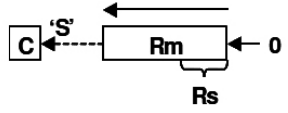
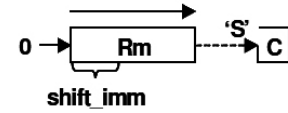
PSR polja (barem jedan sufix)		
Sufiks	Značenje	
c	Control field mask byte	PSR[7:0]
f	Flags field mask byte	PSR[31:24]
s	Status field mask byte	PSR[23:16]
x	Extension field mask byte	PSR[15:8]

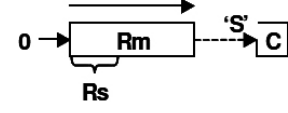
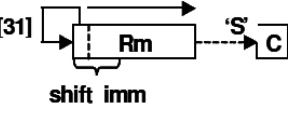
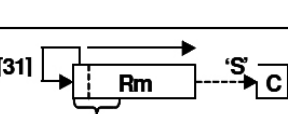

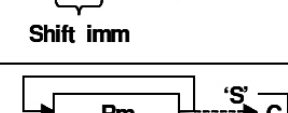
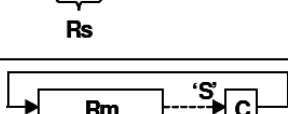
Način adresiranja 2 - Word and Unsigned Byte Data Transfer			
Pre-indexed	Immediate offset	[Rn, #+/-<immed_12>]{!}	
	Zero offset	[Rn]	Equivalent to [Rn,#0]
	Register offset	[Rn, +/-Rm]{!}	
	Scaled register offset	[Rn, +/-Rm, LSL #<immed_5>]{!}	Allowed shifts 0-31
		[Rn, +/-Rm, LSR #<immed_5>]{!}	Allowed shifts 1-32
		[Rn, +/-Rm, ASR #<immed_5>]{!}	Allowed shifts 1-32
Post-indexed	Immediate offset	[Rn, #+/-<immed_12>	
	Register offset	[Rn], +/-Rm	
	Scaled register offset	[Rn], +/-Rm, LSL #<immed_5>	Allowed shifts 0-31
		[Rn], +/-Rm, LSR #<immed_5>	Allowed shifts 1-32
		[Rn], +/-Rm, ASR #<immed_5>	Allowed shifts 1-32
		[Rn], +/-Rm, ROR #<immed_5>	Allowed shifts 1-31

Način adresiranja 3 - Halfword and Signed Byte Data Transfer			
Pre-indexed	Immediate offset	[Rn, #+/-<immed_8>]{!}	
	Zero offset	[Rn]	Equivalent to [Rn,#0]
	Register	[Rn, +/-Rm]{!}	
Post-indexed	Immediate offset	[Rn], #+/-<immed_8>	
	Register	[Rn], +/-Rm	

Način adresiranja 4 - Multiple Data Transfer		
<b>Block load</b>		<b>Stack pop</b>
IA	Increment After	FD Full Descending
IB	Increment Before	ED Empty Descending
DA	Decrement After	FA Full Ascending
DB	Decrement Before	EA Empty Ascending
<b>Block store</b>		<b>Stack push</b>
IA	Increment After	EA Empty Ascending
IB	Increment Before	FA Full Ascending
DA	Decrement After	ED Empty Descending
DB	Decrement Before	FD Full Descending

Operand 2		
Immediate value	#<immed_8r>	
Logical shift left immediate	Rm, LSL #<immed_5>	Allowed shifts 0-31
Logical shift right immediate	Rm, LSR #<immed_5>	Allowed shifts 1-32
Arithmetic shift right immediate	Rm, ASR #<immed_5>	Allowed shifts 1-32
Rotate right immediate	Rm, ROR #<immed_5>	Allowed shifts 1-31
Register	Rm	
Rotate right extended	Rm, RRX	
Logical shift left register	Rm, LSL Rs	
Logical shift right register	Rm, LSR Rs	
Arithmetic shift right register	Rm, ASR Rs	
Rotate right register	Rm, ROR Rs	

Operand	Adresiranje	Objašnjenje
#<immediate>	Neposredno	<immediate> = <immed_8> rotirano udesno za (2 * <rotate_imm>)
<Rm>	Registrarsko	
<Rm>, LSL #<shift_imm>	Registrarsko s neposrednim logičkim posmakom u lijevo	
<Rm>, LSL <Rs>	Registrarsko s registrarskim logičkim posmakom u lijevo	
<Rm>, LSR #<shift_imm>	Registrarsko s neposrednim logičkim posmakom u desno	

Operand	Adresiranje	Objašnjenje
<Rm>, LSR <Rs>	Registrarsko s registrarskim logičkim posmakom u desno	
<Rm>, ASR #<shift_imm>	Registrarsko s neposrednim aritmetičkim posmakom u desno	
<Rm>, ASR <Rs>	Registrarsko s registrarskim aritmetičkim posmakom u desno	
<Rm>, ROR #<shift_imm>	Registrarsko s neposrednim rotiranjem u desno	
<Rm>, ROR <Rs>	Registrarsko s registrarskim rotiranjem u desno	
<Rm>, RRX	Registrarsko s proširenim rotiranjem u desno	

## NAREDBE PROCESORA ARM

Operacija	Opis (engl.)	§	Kod naredbe	Zastavice	Djelovanje
Move	Move		MOV(cond){S} Rd, <Oprnd2>	N Z C	Rd := Oprnd2
	NOT		MVN(cond){S} Rd, <Oprnd2>	N Z C	Rd := 0xFFFFFFFF EOR Oprnd2
	SPSR to register	3	MRS(cond) Rd, SPSR		Rd := SPSR
	CPSR to register	3	MRS(cond) Rd, CPSR		Rd := CPSR
	register to SPSR	3	MSR(cond) SPSR <fields>, Rm		SPSR := Rm (selected bytes only)
	register to CPSR	3	MSR(cond) CPSR <fields>, Rm		CPSR := Rm (selected bytes only)
	immediate to SPSR	3	MSR(cond) SPSR <fields> #immed_8r		SPSR := immed_8r (selected bytes only)
	immediate to CPSR	3	MSR(cond) CPSR <fields> #immed_8r		CPSR := immed_8r (selected bytes only)
Arithmetic	Add		ADD(cond){S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn + Oprnd2
	with carry		ADC(cond){S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn + Oprnd2 + Carry
	Subtract		SUB(cond){S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn - Oprnd2
	with carry		SBC(cond){S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Rn - Oprnd2 - NOT(Carry)
	reverse subtract		RSB(cond){S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Oprnd2 - Rn
	reverse subtract with carry		RSC(cond){S} Rd, Rn, <Oprnd2>	N Z C V	Rd := Oprnd2 - Rn - NOT(Carry)
	Multiply	2	MUL(cond){S} Rd, Rm, Rs	N Z C*	Rd := (Rm * Rs)[31:0]
	accumulate	2	MLA(cond){S} Rd, Rm, Rs, Rn	N Z C*	Rd := ((Rm * Rs) + Rn)[31:0]
	unsigned long	M	UMULL(cond){S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := unsigned(Rm * Rs)
	unsigned accumulate long	M	UMLAL(cond){S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := unsigned(RdHi, RdLo + Rm * Rs)
	signed long	M	SMULL(cond){S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := signed(Rm * Rs)
	signed accumulate long	M	SMALAL(cond){S} RdLo, RdHi, Rm, Rs	N Z C* V*	RdHi, RdLo := signed(RdHi, RdLo + Rm * Rs)
Logical	Count leading zeroes	5	CLZ(cond) Rd, Rm	N Z C*	Rd := number of leading zeroes in Rm
	Test		TST(cond) Rn, <Oprnd2>	N Z C	Update CPSR flags on Rn AND Oprnd2
	Test equivalence		TEQ(cond) Rn, <Oprnd2>	N Z C	Update CPSR flags on Rn EOR Oprnd2
	AND		AND(cond){S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn AND Oprnd2
	EOR		EOR(cond){S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn EOR Oprnd2
	ORR		ORR(cond){S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn OR Oprnd2
	Bit Clear		BIC(cond){S} Rd, Rn, <Oprnd2>	N Z C	Rd := Rn AND NOT Oprnd2
Compare	No operation		NOP		R0 := R0
	Compare		CMP(cond) Rn, <Oprnd2>	N Z C V	Update CPSR flags on Rn - Oprnd2
	negative		CMN(cond) Rn, <Oprnd2>	N Z C V	Update CPSR flags on Rn + Oprnd2

Operacija	Opis (engl.)	§	Kod naredbe	Djelovanje	Napomene
Branch	Branch		B(cond) label	R15 := label	label must be within ±32Mb of current instruction.
	with link		BL(cond) label	R14 := R15-4, R15 := label	label must be within ±32Mb of current instruction.
Load	Word		LDR(cond) Rd, <a_mode2>	Rd := [address]	
	branch (and exchange)		LDR(cond) R15, <a_mode2>	R15 := [address][31:1]	
	Byte		LDR(cond)B Rd, <a_mode2>	Rd := ZeroExtend(byte from address)	
	signed	4	LDR(cond)SB Rd, <a_mode3>	Rd := SignExtend(byte from address)	
	Halfword	4	LDR(cond)H Rd, <a_mode3>	Rd := ZeroExtend(halfword from address)	
Load multiple	signed	4	LDR(cond)SH Rd, <a_mode3>	Rd := SignExtend(halfword from address)	
	Pop, or Block data load		LDM(cond)<a_mode4> Rd(!), <reglist-pc>	Load list of registers from [Rd]	
	return (and exchange) and restore CPSR		LDM(cond)<a_mode4> Rd(!), <reglist+pc>	Load registers, R15 := [address][31:1] Load registers, branch, CPSR := SPSR	Use from exception modes only.
Store	Word		STR(cond) Rd, <a_mode2>	[address] := Rd	
	Byte		STR(cond)B Rd, <a_mode2>	[address][7:0] := Rd[7:0]	
	Halfword	4	STR(cond)H Rd, <a_mode3>	[address][15:0] := Rd[15:0]	
Store multiple	Push, or Block data store		STM(cond)<a_mode4> Rd(!), <reglist>	Store list of registers to [Rd]	
	User mode registers		STM(cond)<a_mode4> Rd(!), <reglist>^	Store list of User mode registers to [Rd]	Use from privileged modes only.
Software interrupt			SWI(cond) <immed_24>	Software interrupt processor exception	24-bit value encoded in instruction.

## GPIO

Adresa	Naziv registra	Opis
GPIO_bazna_adr	GPIOPADR	8-bitni registar podataka, vrata A
GPIO_bazna_adr + 0x4	GPIOBDR	8-bitni registar podataka, vrata B
GPIO_bazna_adr + 0x8	GPIOADDR	8-bitni registar smjera podataka za vrata A
GPIO_bazna_adr + 0xC	GPIOBDDR	8-bitni registar smjera podataka za vrata B

## Opis registara

## GPIOPADR (GPIO Port A Data Register)

GPIOPADR je 8-bitni registar podataka za vrata A. Podatci upisani u ovaj registar postavljeni su na izlazne priključke ako je pripadni bit u registru smjera podataka (GPIOADDR) postavljen u logičku jedinicu. Čitanje ovog registra vraća trenutno stanje priključaka koji su programirani kao ulazni. Za priključke programirane kao izlazne, vraća se vrijednost zadnjeg podatka upisanog u ovaj registar.

## GPIOBDR (GPIO Port B Data Register)

GPIOBDR je 8-bitni registar podataka za vrata B. Podatci upisani u ovaj registar postavljeni su na izlazne priključke ako je pripadni bit u registru smjera podataka (GPIOBDDR) postavljen u logičku nulu (ovo je suprotno od vrata A). Čitanje ovog registra vraća trenutno stanje priključaka koji su programirani kao ulazni. Za priključke programirane kao izlazne, vraća se vrijednost zadnjeg podatka upisanog u ovaj registar.

## GPIOADDR (GPIO Port A Data Direction Register)

GPIOADDR je 8-bitni registar smjera podataka za vrata A. Bit postavljen u logičku nulu u ovom registru konfigurira odgovarajući priključak od vrata A kao izlazni. Postavljanje bita u nulu konfigurira odgovarajući priključak vrata A kao ulazni.

## GPIOBDDR (GPIO Port B Data Direction Register)

GPIOBDDR je 8-bitni registar smjera podataka za vrata B. Bit postavljen u logičku nulu u ovom registru konfigurira odgovarajući priključak od vrata B kao izlazni. Postavljanje bita u jedinicu konfigurira odgovarajući priključak vrata B kao ulazni.

## Početna vrijednost registara

Svi registri unutar GPIO nakon inicijalizacije (resetiranja) postavljaju se u logičku nulu. Ovine se inicijalno vrata A postavljaju kao ulazna, a vrata B kao izlazna. Sadržaji obaju registara podataka su nula.

## RTC

Adresa	Naziv registra	Opis
RTC_bazna_adr	RTCDR	32-bitni registar podataka (može se samo čitati)
RTC_bazna_adr + 0x4	RTCMR	32-bitni registar usporedbe
RTC_bazna_adr + 0x8	RTCSTAT/RTCEOI	1-bitni registar stanja prekida (ako se čita) / 0-bitni registar za brisanje prekida (ako se piše)
RTC_bazna_adr + 0xC	RTCCLR	32-bitni registar za punjenje brojila
RTC_bazna_adr + 0x10	RTCCR	1-bitni upravljački registar

## Opis registara

## RTCDR (Real Time Clock Data Register)

RTCDR je 32-bitni registar podataka. Čitanjem ovog registra dobiva se trenutna vrijednost brojila. Pisanje u ovaj registar nije dozvoljeno.

## RTCMR (Real Time Clock Match Register)

RTCMR je 32-bitni registar usporedbe. Upisivanjem podatka u ovaj registar postavlja se nova vrijednost koja služi za usporedbu s brojilom. Čitanjem ovog registra dobiva se zadnja vrijednost upisana u registar usporedbe.

## RTCSTAT/RTCEOI (Real Time Clock Interrupt STATus Register/Real Time Clock Interrupt Clear Register)

RTCSTAT/RTCEOI je virtualni registar bez fizičkog sklopovlja za pohranjivanje podataka. Pisanjem bilo kojeg podatka na ovu adresu čisti se prekidni signal RTCINTR i pripadni registar. Čitanjem s ove adrese dobiva se podatak koji na bitu 0 (najniži bit) ima trenutnu vrijednost RTCINTR. Ako je bit 0 postavljen na jedinicu, to znači da je prekidni signal aktivan.

## RTCLR (Real Time Clock Load Register)

RTCLR je 32-bitni registar koji služi za upis vrijednosti brojila ili čitanje zadnje vrijednosti koja je upisana. Pisanjem u ovaj registar započinje proces pisanja nove vrijednosti u brojilo. Pisanje se ne izvodi trenutno nego na prvi sljedeći rastući brid na ulazu CLK1HZ.

## RTCCR (Real Time Clock Control Register)

RTCCR je 1-bitni upravljački registar kojim programer može omogućiti ili onemogućiti generiranje prekida. Ako se na bit 0 (najniži bit) ovog registra upiše logička nula, tada se RTC-u onemogućuje generiranje prekida. Ako se upiše jedinica, tada se omogućuje generiranje prekida. Čitanjem ovog registra na bitu 0 dobiva se zadnja upisana vrijednost prekidnog bita. Ostali bitovi u ovom registru ne postoje.

## Početna vrijednost registara

Svi registri unutar sklopa RTC nakon inicijalizacije (resetiranja) postavljaju se u logičku nulu.