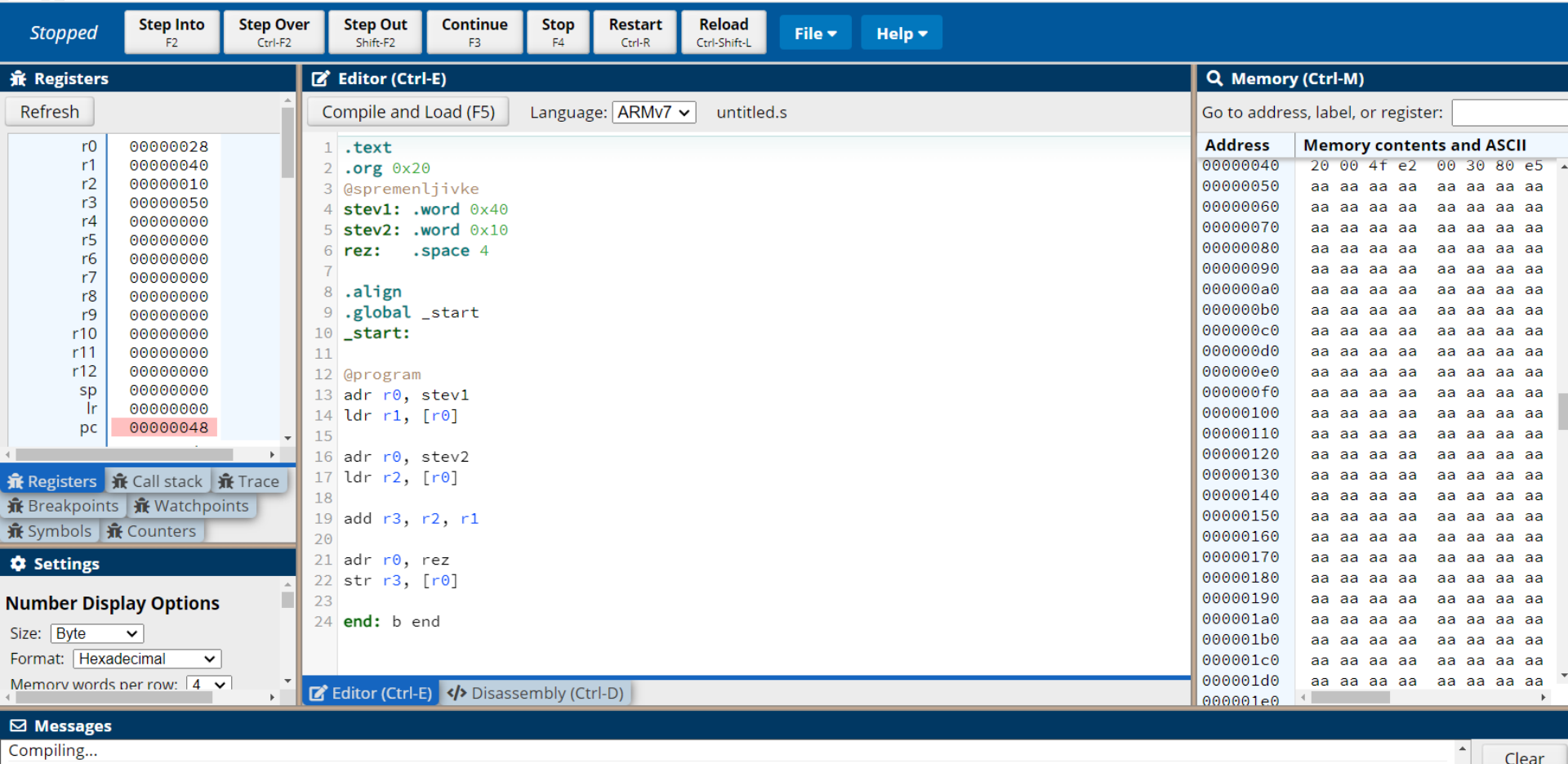
ARM zbirnik – OR nadgradnja

Arhitektura in
programiranje v zbirniku

Spletni simulator cpulator

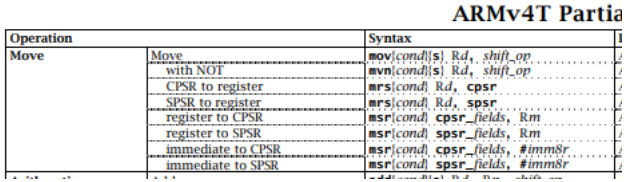
- **CPUlator ARMv7 System Simulator (01xz.net)**



Začetni projekt OR

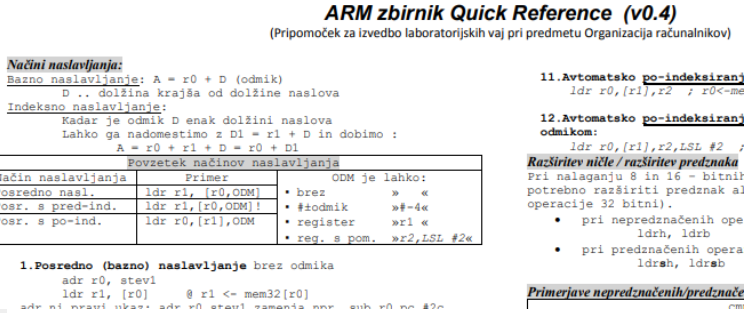
Pripomočki

ARM Ref. in OR QuickRef



ARMv4T Partia

Operation		Syntax
Move	Move	mov{cond}{s} Rd, shift_op
	with NOT	mvn{cond}{s} Rd, shift_op
	CPSR to register	mrs{cond} Rd, cpsr
	SPSR to register	mrs{cond} Rd, spsr
	register to CPSR	msr{cond} cpsr_fields, Rm
	register to SPSR	msr{cond} spsr_fields, Rm
	immediate to CPSR	msr{cond} cpsr_fields, #imm8r
	immediate to SPSR	msr{cond} spsr_fields, #imm8r



ARM zbirnik Quick Reference (v0.4)

(Pripomoček za izvedbo laboratorijskih vaj pri predmetu Organizacija računalnikov)

Načini naslavljanja:

Bazno naslavljanje: A = r0 + D (odmik)
D .. dolžina krajša od dolžine naslova

Indeksno naslavljanje:
Kadar je odmik D enak dolžini naslova
Lahko ga nadomestimo z D1 = r1 + D in dobimo :
A = r0 + r1 + D = r0 + D1

Povzetek načinov naslavljanja

Način naslavljanja	Primer	ODM je lahko:
Posredno nasl.	ldr r1, [r0,ODM]	• brez » «
Posr. s pred-ind.	ldr r1,[r0,ODM]!	• #±odmik »#-4«
Posr. s po-ind.	ldr r0,[r1],ODM	• register »r1 «
		• reg. s pom. »r2,LSL #2«

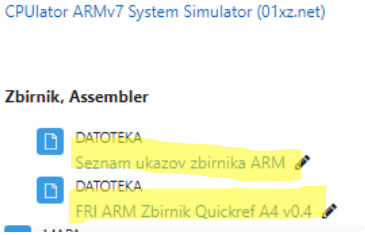
1.Posredno (bazno) naslavljanje brez odmika
adr r0, stev1
ldr r1, [r0] @ r1 <- mem32[r0]

11.Avtomatsko po-indeksiranj
ldr r0,[r1],r2 ; r0<-me

12.Avtomatsko po-indeksiranj
odmikom:
ldr r0,[r1],r2,LSL #2 ;

Razširitev ničle / razširitev predznaka
Pri nalaganju 8 in 16 - bitnih
potrebno razširiti predznak al
operacije 32 bitni).
- pri nepredznačenih ope
ldrh, ldrb
- pri predznačenih opera
ldrsh, ldrsb

Primerjave nepredznačenih/predznače



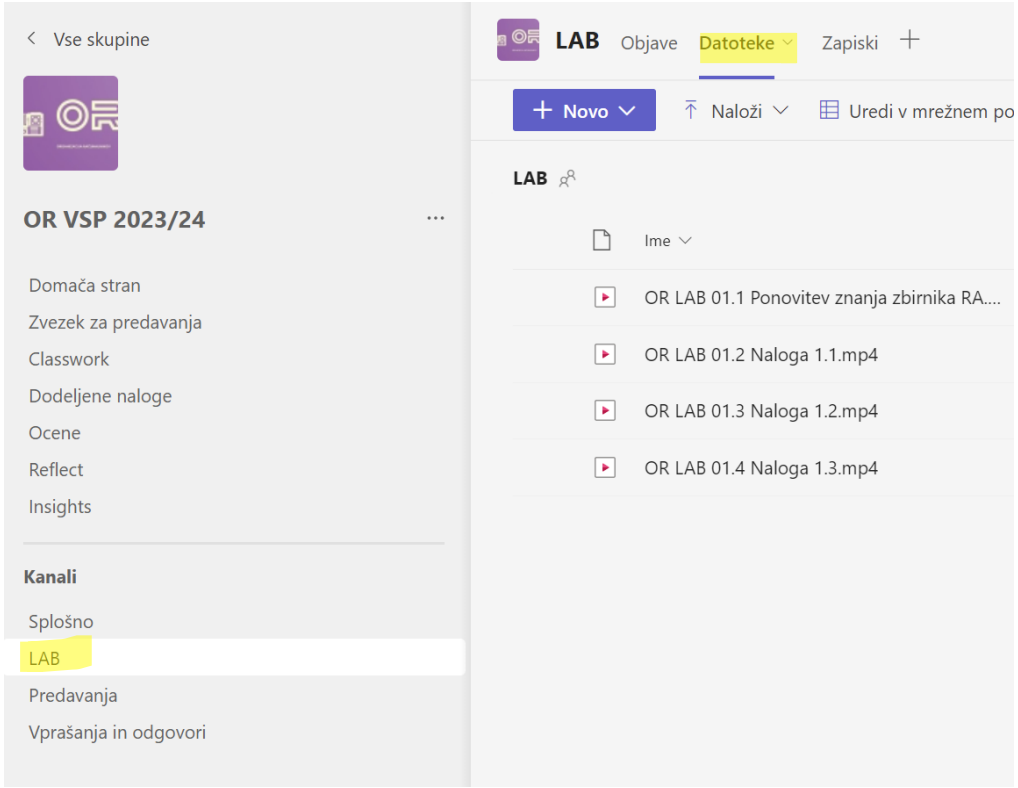
CPUlator ARMv7 System Simulator (01xz.net)

Zbirnik, Assembler

DATOTEKA
Seznam ukazov zbirnika ARM

DATOTEKA
FRI ARM Zbirnik Quickref A4 v0.4

Video posnetki



Vse skupine

OR VSP 2023/24

Domača stran
Zvezek za predavanja
Classwork
Dodeljene naloge
Ocene
Reflect
Insights

Kanali

Splošno
LAB
Predavanja
Vprašanja in odgovori

LAB Objave Datoteke Zapiski

Novo Naloži Uredi v mrežnem po

LAB

Ime

OR LAB 01.1 Ponovitev znanja zbirnika RA....
OR LAB 01.2 Naloga 1.1.mp4
OR LAB 01.3 Naloga 1.2.mp4
OR LAB 01.4 Naloga 1.3.mp4

Začetni projekt OR

Logični ukazi (delo z določenimi biti)

`and r1, r2, r3 @brisanje z ničlo v maski določenih bitov`

r2	00000000	00000000	00000000	01010011
and r3	11111111	11111111	11111111	11001010
=r1	00000000	00000000	00000000	01000010

`bic r1, r2, r3 @brisanje z enico v maski določenih bitov`

r2	00000000	00000000	00000000	01010011
bic r3	11111111	11111111	11111111	11001010
=r1	00000000	00000000	00000000	00010001

`orr r1, r2, r3 @postavljanje z enico v maski določenih bitov`

r2	00000000	00000000	00000000	01010011
or r3	11111111	11111111	11111111	11001010
=r1	11111111	11111111	11111111	11011011

`eor r1, r2, r3 @invertiranje z enico v maski določenih bitov`

r2	00000000	00000000	00000000	01010011
eor r3	11111111	11111111	11111111	11001010
=r1	11111111	11111111	11111111	10011001

Logični ukazi (preverjanje stanja določenih bitov)

- **Preverjanje stanja enega bita (določen je z enico v maski)**

```
tst r1, r2 @zastavice postavi glede na r1 AND r2
```

r1	00000000	00000000	00000000	01010011	
tst r2	00000000	00000000	00000000	00000100	
=	00000000	00000000	00000000	00000000	➔ Z=1

r1	00000000	00000000	00000000	01010011	
tst r2	00000000	00000000	00000000	00000010	
=	00000000	00000000	00000000	00000010	➔ Z=0

- **Preverjanje stanja večih bitov:**
  - Najprej izločimo bite, ki nas zanimajo: and
  - Primerjamo z želenim stanjem bitov (bite, ki nas ne zanimajo, primerjamo z 0)

Zgled:

```
@preveri, da je bit7 v r1 enak 0 in bit 2 v r1 enak 1
and r2, r1, #0x84      @0x84 = 00…010000100 => r2 = 00..00?0000?00
cmp r2, #0x04          @0x04 = 00…000000100; ustreza, če Z=1
```

Aritmetično-logični ukazi, seznam

- **Aritmetični ukazi:**

```
add r0, r1, r2    ; r0 <- r1 + r2
adc r0, r1, r2    ; r0 <- r1 + r2 + C       (add with C)
sub r0, r1, r2    ; r0 <- r1 - r2
sbc r0, r1, r2    ; r0 <- r1 - r2 + C - 1   (-not(C)=-(1-C)= C-1
rsb r0, r1, r2    ; r0 <- r2 - r1           (reverse subtract)
rsc r0, r1, r2    ; r0 <- r2 - r1 + C - 1   (rev. sub -not(C) )
```

- **Logični ukazi:**

```
and r0, r1, r2    ; r0 <- r1 AND r2
orr r0, r1, r2    ; r0 <- r1 OR r2
eor r0, r1, r2    ; r0 <- r1 XOR r2
bic r0, r1, r2    ; r0 <- r1 AND NOT r2
```

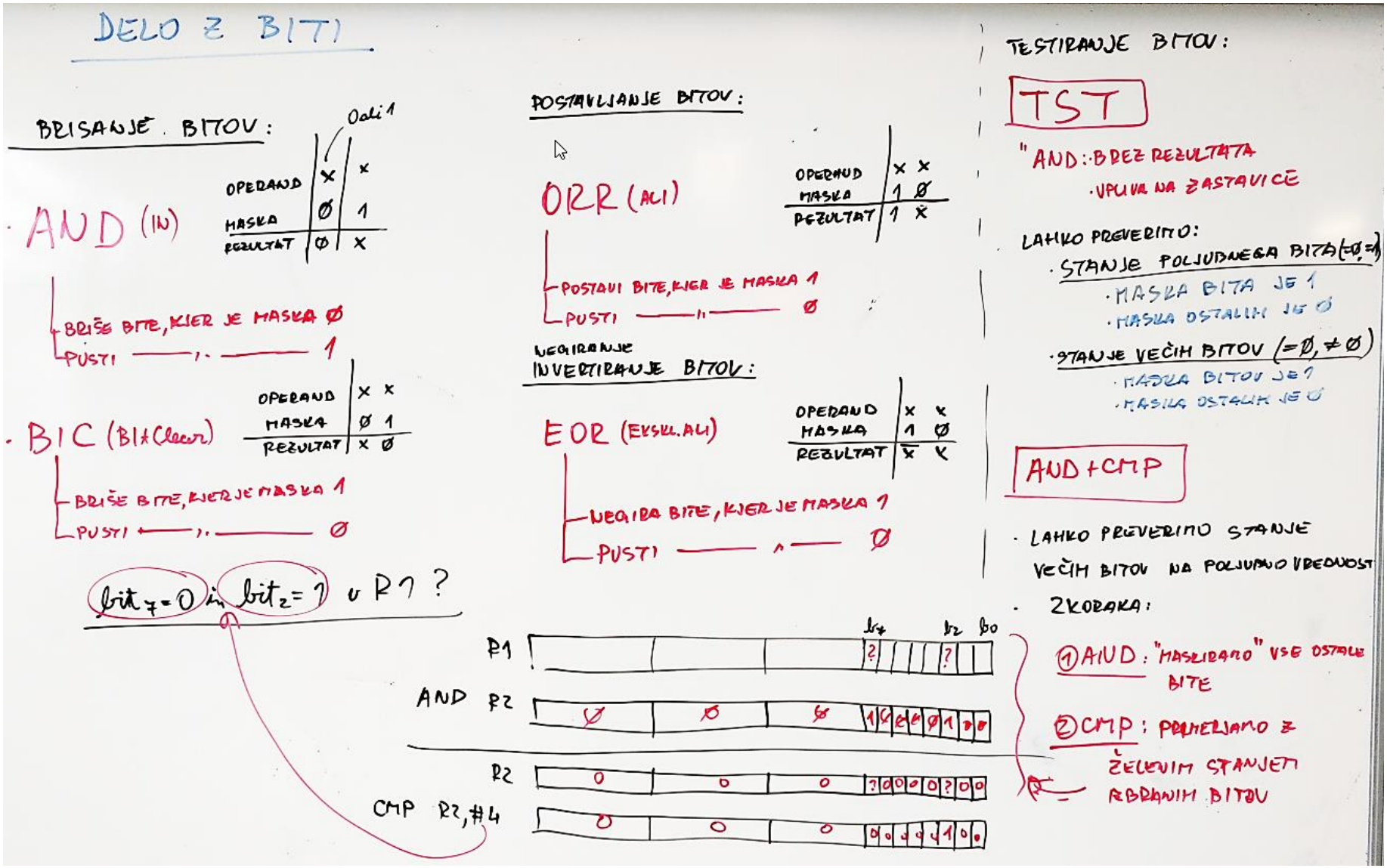
- **Prenos med registri:**

```
mov r0, r2        ; r0 <- r2
mvn r0, r2        ; r0 <- NOT r2
```

- **Primerjave:**

```
cmp r1, r2        ; set CPSR flags on r1 - r2
cmn r1, r2        ; set CPSR flags on r1 + r2
tst r1, r2        ; set CPSR flags on r1 AND r2
teq r1, r2        ; set CPSR flags on r1 XOR r2      (equivalence test)
```

LAB -1 : Tabla


DELO Z BITI
BRISANJE BITOV:
Oali 1
AND (IN)
OPERAND x x
MASKA Ø 1
REZULTAT Ø x
BRIŠE BITE, KJER JE MASKA Ø
PUSTI 1
BIC (Bit Clear)
OPERAND x x
MASKA Ø 1
REZULTAT x Ø
BRIŠE BITE, KJER JE MASKA 1
PUSTI Ø
POSTAVLJANJE BITOV:
ORR (ALI)
OPERAND x x
MASKA 1 Ø
REZULTAT 1 x
POSTAVI BITE, KJER JE MASKA 1
PUSTI Ø
NEGIRANJE
INVERTIRANJE BITOV:
EOR (EKSKL. ALI)
OPERAND x x
MASKA 1 Ø
REZULTAT x x
NEGIRA BITE, KJER JE MASKA 1
PUSTI Ø
bit7 = 0 in bit2 = 1 v R1 ?
R1
AND R2
R2
CMP R2, #4
TESTIRANJE BITOV:
TST
" AND: BREZ REZULTATA
· VPLIVA NA ZASTAVICE
LAHKO PREVERIMO:
· STANJE POLJUBNEGA BITA (=Ø, ≠Ø)
· MASKA BITA JE 1
· MASKA OSTALIM JE Ø
· STANJE VEČIH BITOV (=Ø, ≠Ø)
· MASKA BITOV JE 1
· MASKA OSTALIM JE Ø
AND + CMP
· LAHKO PREVERIMO STANJE VEČIH BITOV NA POLJUBNO VREDNOST
· 2 KORAKA:
① AND: "MASKIRANO" VSE OSTALE BITE
② CMP: PRIMERJAMO Z ŽELENIM STANJEM OBRANIH BITOV