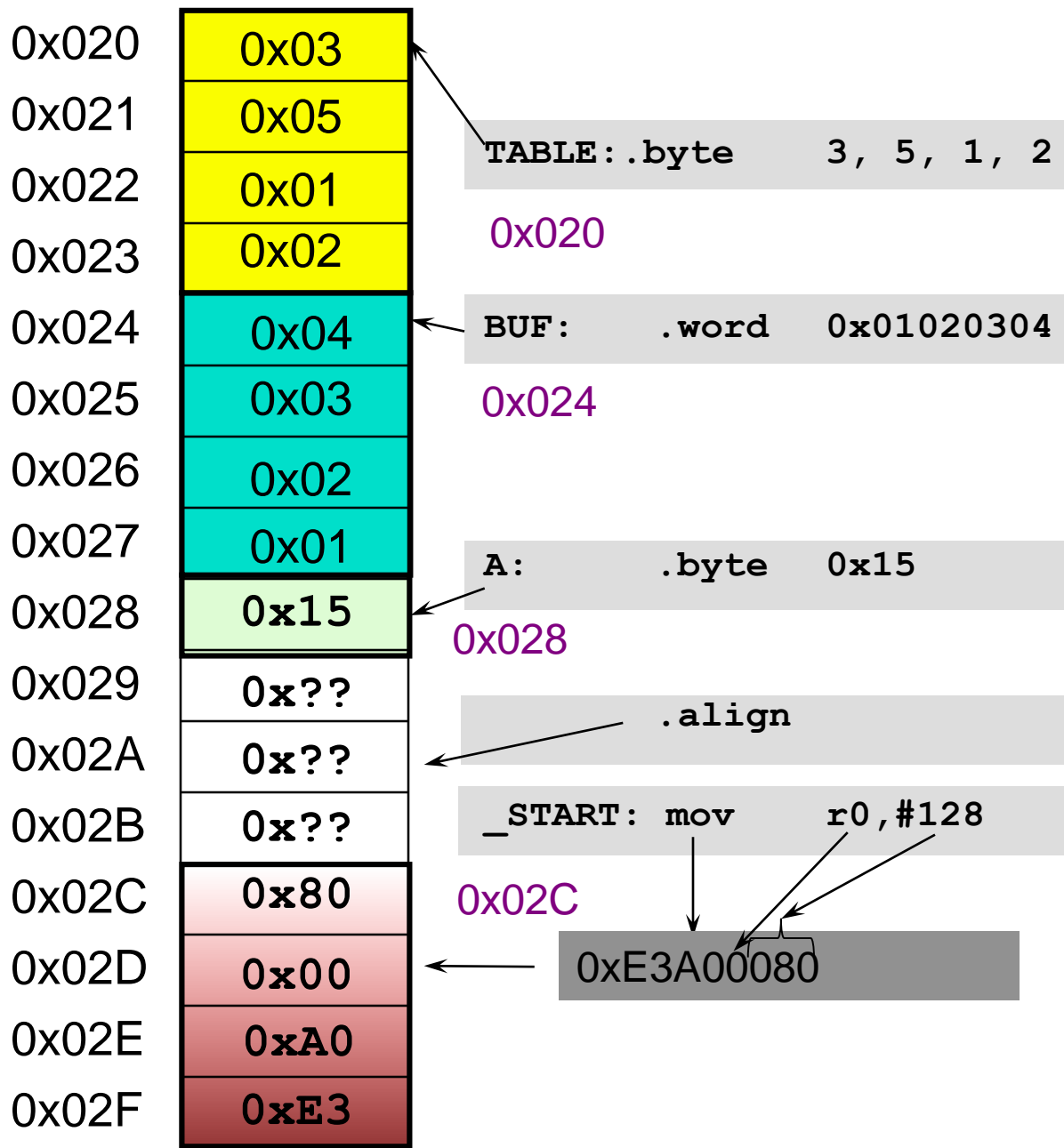


Povzetek – prevajanje (psevdoukazi, ukazi)



| | |
|----------------|-------|
| ZBIRNIK | |
| Števec lokacij | |
| 0x030 | |
| Tabela oznak | |
| TABLE | 0x020 |
| BUF | 0x024 |
| A | 0x028 |
| _START | 0x02C |
| | |
| | |
| | |
| | |
| | |

Ukazi load/store – načini naslavljanja

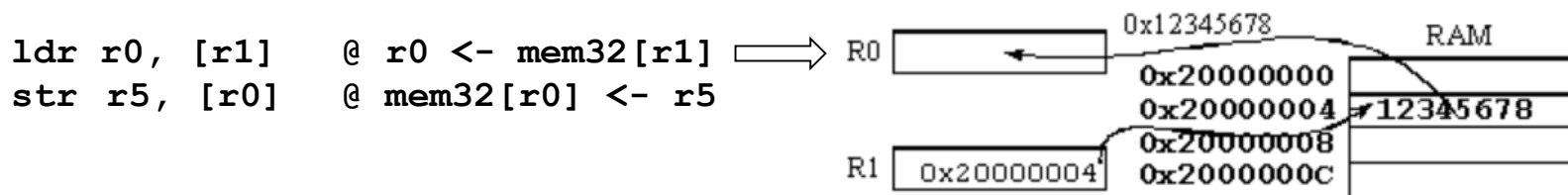
1. Posredno naslavljanje – bazno naslavljanje brez odmika

Dostop do operanda v dveh korakih :

a) naslov spremenljivke najprej naložimo v bazni register z:

```
adr r1, stev1
```

b) nato uporabimo ukaz load/store oblike za dostop do operanda:



Opomba:

adr je psevdo ukaz. Prevajalnik ga nadomesti z ALE ukazom, ki izračuna naslov spremenljivke s pomočjo R15 (PC) in konstante.

Primer:

```
adr r0, stev1 prevajalnik nadomesti npr. s sub r0, pc, #2c
```

Ukazi load/store – načini naslavljanja

Zgledi za bazno naslavljanje brez odmika

32-bitni operandi

```
adr r1, VAR1
ldr r0, [r1]
str r0, [r1]
```

```
@ r1 <- naslov sprem. VAR1
@ r0 <- mem32[r1]
@ mem32[r1] <- r0
```

16-bitni operandi

```
adr r1, VAR2
ldr(s)h r0, [r1]
strh ↑ r0, [r1]
```

```
@ r1 <- naslov sprem. VAR2
@ r0 <- mem16[r1]
@ mem16[r1] <- r0[b0..b15]
```

8-bitni operandi

```
adr r1, VAR3
ldr(s)b r0, [r1]
strb ↑ r0, [r1]
```

```
@ r1 <- naslov sprem. VAR3
@ r0 <- mem8[r1]
@ mem8[r1] <- r0[b0..b7]
```

s..operand je predznačeno število

Load/store – načini naslavljanja

2. Posredno naslavljanje – bazno naslavljanje s takojšnjim odmikom (preindex with immediate offset):

```
ldr r0,[r1, #n12]      @ r0<-mem32[r1+n12]
str r0,[r1, #n12]      @ mem32[r1+n12]<-r0
strb r0,[r1, #n12]     @ mem8[r1+n12]<-r0[b0..b7]
```

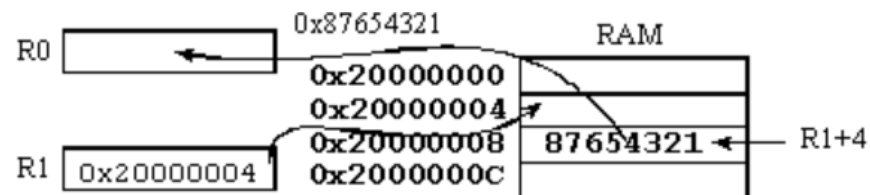
```
ldr(s)b r0,[r1, #n8]   @ r0<-mem8[r1+n8]
ldr(s)h r0,[r1, #n8]   @ r0<-mem16[r1+n8]
strh r0,[r1, #n8]     @ mem16[r1+n8]<-r0[b0..b15]
```

n12 – 12-bitni predznačen odmik

n8 – 8-bitni predznačen odmik

Zgledi:

```
ldr r0, [r1, #4]      @ r0 <- mem32[r1 + 4]
ldr r5, [r0, #-20]    @ r5 <- mem32[r0 - 20]
                      @ v r0 mora biti ustrezen naslov!!!
strb r7, [r2,#10]     @ mem8[r2 + 10] <- r7[b0..b7]
                      @ v r2 mora biti ustrezen naslov!!!
```

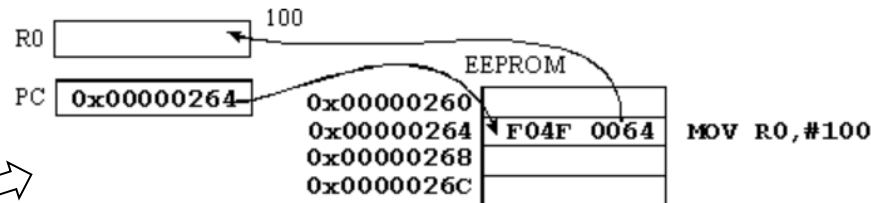


Naslov operanda je vsota **baznega registra** in **predznačenega odmika**

Aritmetično-logični ukazi

3. Takojšnje naslavljanje

`mov r0, #100` @ `r0 <- 100`
`add r2, r7, #0x20` @ `r2 <- r7 + 32`
`sub r5, r5, #1` @ `r5 <- r5 - 1`



Takojšnji operand = $(0..255) * 2^{2*(0..12)}$

32-bitni takojšnji operand je 8-bitno število, ki ga lahko še krožno pomaknemo za sodo število mest znotraj 32-bitne vsebine. Takojšnji operand torej ni poljuben! Tvor ga prevajalnik - če ga ne more, nas opozori.

Takojšnji operand je del ukaza, torej mora biti v času prevajanja iz zbirnega v strojni jezik že znan. Takojšnji operandi so **konstante**.

Aritmetično-logični ukazi (takojšnje naslavljanje)

Zgledi

Pravilni takojšnji operandi:

[illegible]

Napačni takojšnji operandi:

```
mov r1, #257 @ r1 <- 0b000000000000000000000000100000001
add r7, r3, #65535 r7 <- r3 + 0b000000000000000000111111111111111
```

Takojšnji operand je **nepredznačeno 8-bitno število**, ki ga lahko še krožno pomaknemo za 2^n bitov v levo, kjer je n lahko med 0 in 12.

Aritmetično-logični ukazi

4. Neposredno registrsko naslavljanje

- za računanje z registri in prepisovanje vrednosti iz enega registra v drugega.

```
add r2, r7, r12  
sub r4, r5, r1  
mov r1, r4
```

Aritmetično-logični ukazi, seznam

• Aritmetični ukazi:

| | | |
|-----------------------------|---|------------------------|
| <code>add r0, r1, r2</code> | <code>@ r0 <- r1 + r2</code> | |
| <code>adc r0, r1, r2</code> | <code>@ r0 <- r1 + r2 + C</code> | (add with C) |
| <code>sub r0, r1, r2</code> | <code>@ r0 <- r1 - r2</code> | |
| <code>sbc r0, r1, r2</code> | <code>@ r0 <- r1 - r2 + C - 1</code> | (-not(C)=- (1-C)= C-1) |
| <code>rsb r0, r1, r2</code> | <code>@ r0 <- r2 - r1</code> | (reverse subtract) |
| <code>rsc r0, r1, r2</code> | <code>@ r0 <- r2 - r1 + C - 1</code> | (rev. sub -not(C)) |

• Logični ukazi:

| | |
|-----------------------------|---------------------------------------|
| <code>and r0, r1, r2</code> | <code>@ r0 <- r1 AND r2</code> |
| <code>orr r0, r1, r2</code> | <code>@ r0 <- r1 OR r2</code> |
| <code>eor r0, r1, r2</code> | <code>@ r0 <- r1 XOR r2</code> |
| <code>bic r0, r1, r2</code> | <code>@ r0 <- r1 AND NOT r2</code> |

• Prenos med registri:

| | |
|-------------------------|--------------------------------|
| <code>mov r0, r2</code> | <code>@ r0 <- r2</code> |
| <code>mvn r0, r2</code> | <code>@ r0 <- NOT r2</code> |

• Primerjave:

| | | |
|-------------------------|--|--------------------|
| <code>cmp r1, r2</code> | <code>@ set CPSR flags on r1 - r2</code> | |
| <code>cmn r1, r2</code> | <code>@ set CPSR flags on r1 + r2</code> | |
| <code>tst r1, r2</code> | <code>@ set CPSR flags on r1 AND r2</code> | |
| <code>teq r1, r2</code> | <code>@ set CPSR flags on r1 XOR r2</code> | (equivalence test) |