

ARM

PROGRAMIRANJE V  
ZBIRNEM JEZIKU

*1. del*

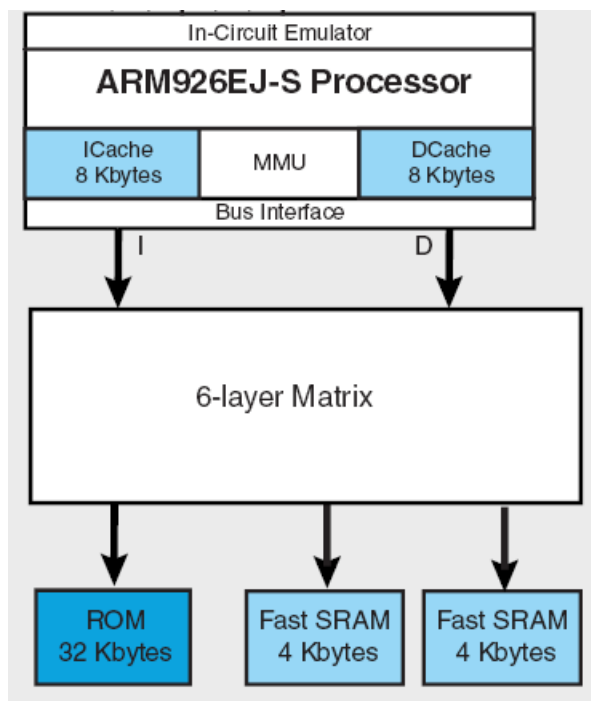
# ARM (Advanced RISC Machine) = RISC?

- + load/store arhitektura
  - + cevovodna zgradba
  - + reduciran nabor ukazov, vsi ukazi 32-bitni
  - + ortogonalen registrski niz. Vsi registri 32-bitni
- 
- veliko načinov naslavljanja
  - veliko formatov ukazov
- 
- nekateri ukazi se izvajajo več kot en cikel (npr. *load/store multiple*) – obstaja nekaj kompleksnejših ukazov, kar omogoča manjšo velikost programov
  - dodaten 16-bitni nabor ukazov Thumb omogoča krajše programe
  - pogojno izvajanje ukazov – ukaz se izvede le, če je stanje zastavic ustrezno.

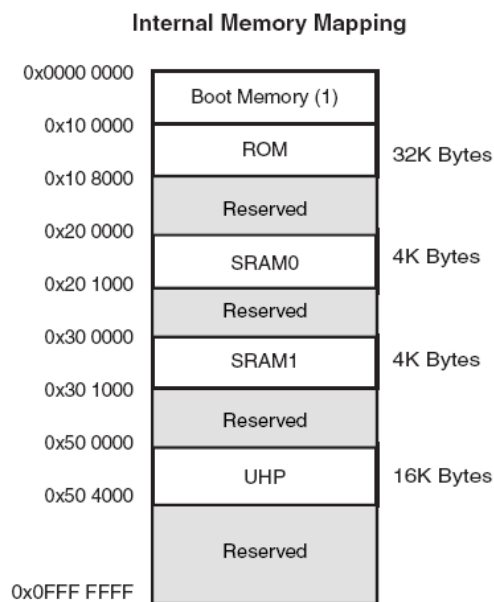
The diagram illustrates the internal architecture of the ARM926EJ-S processor and its system components. Key elements include:

- System Controller:** Manages various system functions like FIQ, IRQ, DRXD, DTXD, PCK0-PCK1, PLLA, PLLB, OSC, WDT, PIT, RC, 4GPREG, OSCSEL, XIN32, XOUT32, SHDN, WKUP, VDDBU, VDDCORE, NRST, PICA, PIOB, PIOC, and RSTC.
- In-Circuit Emulator:** Facilitates JTAG Selection and Boundary Scan.
- ARM926EJ-S Processor:** Features ICache (8 Kbytes), MMU, DCache (8 Kbytes), and a Bus Interface.
- 6-layer Matrix:** Connects the processor to various peripherals.
- Peripherals:**
  - 10/100 Ethernet MAC:** Includes FIFO and DMA.
  - Image Sensor Interface:** Includes DMA.
  - USB OHCI:** Includes DMA.
  - Peripheral Bridge:** Connects to various I/O controllers.
  - 22-channel Peripheral DMA:** Manages data flow between peripherals.
  - EBI (External Bus Interface):** Manages CompactFlash NAND Flash, SDRAM Controller, Static Memory Controller, and ECC Controller.
  - I/O Controllers:** MCI, TWI, UART0-5, SPI0-1, TC0-5, SSC, 4-channel 10-bit ADC, USB Device, and Transceiver.
- Input/Output Signals:** NRST, TD0, TD1, TD2, TD3, TD4, TD5, TD6, TD7, TD8, TD9, TD10, TD11, TD12, TD13, TD14, TD15, TD16, TD17, TD18, TD19, TD20, TD21, TD22, TD23, TD24, TD25, TD26, TD27, TD28, TD29, TD30, TD31, TD32, TD33, TD34, TD35, TD36, TD37, TD38, TD39, TD40, TD41, TD42, TD43, TD44, TD45, TD46, TD47, TD48, TD49, TD50, TD51, TD52, TD53, TD54, TD55, TD56, TD57, TD58, TD59, TD60, TD61, TD62, TD63, TD64, TD65, TD66, TD67, TD68, TD69, TD70, TD71, TD72, TD73, TD74, TD75, TD76, TD77, TD78, TD79, TD80, TD81, TD82, TD83, TD84, TD85, TD86, TD87, TD88, TD89, TD90, TD91, TD92, TD93, TD94, TD95, TD96, TD97, TD98, TD99, TD100, TD101, TD102, TD103, TD104, TD105, TD106, TD107, TD108, TD109, TD110, TD111, TD112, TD113, TD114, TD115, TD116, TD117, TD118, TD119, TD120, TD121, TD122, TD123, TD124, TD125, TD126, TD127, TD128, TD129, TD130, TD131, TD132, TD133, TD134, TD135, TD136, TD137, TD138, TD139, TD140, TD141, TD142, TD143, TD144, TD145, TD146, TD147, TD148, TD149, TD150, TD151, TD152, TD153, TD154, TD155, TD156, TD157, TD158, TD159, TD160, TD161, TD162, TD163, TD164, TD165, TD166, TD167, TD168, TD169, TD170, TD171, TD172, TD173, TD174, TD175, TD176, TD177, TD178, TD179, TD180, TD181, TD182, TD183, TD184, TD185, TD186, TD187, TD188, TD189, TD190, TD191, TD192, TD193, TD194, TD195, TD196, TD197, TD198, TD199, TD200, TD201, TD202, TD203, TD204, TD205, TD206, TD207, TD208, TD209, TD210, TD211, TD212, TD213, TD214, TD215, TD216, TD217, TD218, TD219, TD220, TD221, TD222, TD223, TD224, TD225, TD226, TD227, TD228, TD229, TD230, TD231, TD232, TD233, TD234, TD235, TD236, TD237, TD238, TD239, TD240, TD241, TD242, TD243, TD244, TD245, TD246, TD247, TD248, TD249, TD250, TD251, TD252, TD253, TD254, TD255, TD256, TD257, TD258, TD259, TD260, TD261, TD262, TD263, TD264, TD265, TD266, TD267, TD268, TD269, TD270, TD271, TD272, TD273, TD274, TD275, TD276, TD277, TD278, TD279, TD280, TD281, TD282, TD283, TD284, TD285, TD286, TD287, TD288, TD289, TD290, TD291, TD292, TD293, TD294, TD295, TD296, TD297, TD298, TD299, TD300, TD301, TD302, TD303, TD304, TD305, TD306, TD307, TD308, TD309, TD310, TD311, TD312, TD313, TD314, TD315, TD316, TD317, TD318, TD319, TD320, TD321, TD322, TD323, TD324, TD325, TD326, TD327, TD328, TD329, TD330, TD331, TD332, TD333, TD334, TD335, TD336, TD337, TD338, TD339, TD340, TD341, TD342, TD343, TD344, TD345, TD346, TD347, TD348, TD349, TD350, TD351, TD352, TD353, TD354, TD355, TD356, TD357, TD358, TD359, TD360, TD361, TD362, TD363, TD364, TD365, TD366, TD367, TD368, TD369, TD370, TD371, TD372, TD373, TD374, TD375, TD376, TD377, TD378, TD379, TD380, TD381, TD382, TD383, TD384, TD385, TD386, TD387, TD388, TD389, TD390, TD391, TD392, TD393, TD394, TD395, TD396, TD397, TD398, TD399, TD400, TD401, TD402, TD403, TD404, TD405, TD406, TD407, TD408, TD409, TD410, TD411, TD412, TD413, TD414, TD415, TD416, TD417, TD418, TD419, TD420, TD421, TD422, TD423, TD424, TD425, TD426, TD427, TD428, TD429, TD430, TD431, TD432, TD433, TD434, TD435, TD436, TD437, TD438, TD439, TD440, TD441, TD442, TD443, TD444, TD445, TD446, TD447, TD448, TD449, TD450, TD451, TD452, TD453, TD454, TD455, TD456, TD457, TD458, TD459, TD460, TD461, TD462, TD463, TD464, TD465, TD466, TD467, TD468, TD469, TD470, TD471, TD472, TD473, TD474, TD475, TD476, TD477, TD478, TD479, TD480, TD481, TD482, TD483, TD484, TD485, TD486, TD487, TD488, TD489, TD490, TD491, TD492, TD493, TD494, TD495, TD496, TD497, TD498, TD499, TD500, TD501, TD502, TD503, TD504, TD505, TD506, TD507, TD508, TD509, TD510, TD511, TD512, TD513, TD514, TD515, TD516, TD517, TD518, TD519, TD520, TD521, TD522, TD523, TD524, TD525, TD526, TD527, TD528, TD529, TD530, TD531, TD532, TD533, TD534, TD535, TD536, TD537, TD538, TD539, TD540, TD541, TD542, TD543, TD544, TD545, TD546, TD547, TD548, TD549, TD550, TD551, TD552, TD553, TD554, TD555, TD556, TD557, TD558, TD559, TD560, TD561, TD562, TD563, TD564, TD565, TD566, TD567, TD568, TD569, TD570, TD571, TD572, TD573, TD574, TD575, TD576, TD577, TD578, TD579, TD580, TD581, TD582, TD583, TD584, TD585, TD586, TD587, TD588, TD589, TD590, TD591, TD592, TD593, TD594, TD595, TD596, TD597, TD598, TD599, TD600, TD601, TD602, TD603, TD604, TD605, TD606, TD607, TD608, TD609, TD610, TD611, TD612, TD613, TD614, TD615, TD616, TD617, TD618, TD619, TD620, TD621, TD622, TD623, TD624, TD625, TD626, TD627, TD628, TD629, TD630, TD631, TD632, TD633, TD634, TD635, TD636, TD637, TD638, TD639, TD640, TD641, TD642, TD643, TD644, TD645, TD646, TD647, TD648, TD649, TD650, TD651, TD652, TD653, TD654, TD655, TD656, TD657, TD658, TD659, TD660, TD661, TD662, TD663, TD664, TD665, TD666, TD667, TD668, TD669, TD670, TD671, TD672, TD673, TD674, TD675, TD676, TD677, TD678, TD679, TD680, TD681, TD682, TD683, TD684, TD685, TD686, TD687, TD688, TD689, TD690, TD691, TD692, TD693, TD694, TD695, TD696, TD697, TD698, TD699, TD700, TD701, TD702, TD703, TD704, TD705, TD706, TD707, TD708, TD709, TD710, TD711, TD712, TD713, TD714, TD715, TD716, TD717, TD718, TD719, TD720, TD721, TD722,

# AT91SAM9260



## Shema pomnilniškega prostora



# ARM programski model

- Programski model sestavlja 16 registrov ter statusni register CPSR (Current Program Status Register)
- **Več načinov delovanja, vsak ima nekaj svojih registrov. Vseh registrov je v resnici 36**
- **Kateri registri so vidni je odvisno od načina delovanja procesorja (*processor mode*)**
- **Načine delovanja delimo v dve skupini:**
  - Privilegirani (dovoljena bralni in pisalni dostop do CPSR)
  - Neprivilegirani (dovoljen le bralni dostop do CPSR)

# Programski model – uporabniški način

Uporabniški način (*user mode*):

- edini neprivilegirani način
- v tem načinu se izvajajo uporabniški programi

Programsko je vidnih 17 32-bitnih registrov:  
r0 – r15 ter CPSR

Vidni registri:

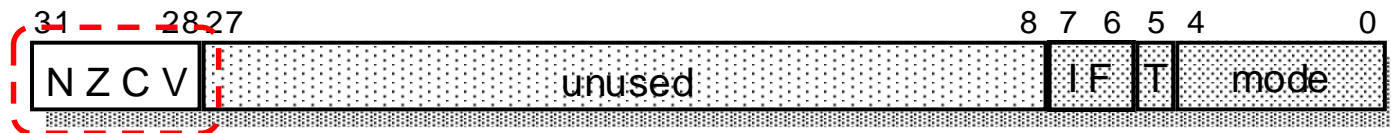
- r0-r12: splošnonamenski (ortogonalni) registri
- r13(sp): skladovni kazalec (*Stack Pointer*)
- r14(lr): povratni naslov (*Link Register*)
- r15(pc): programski števec (*Program Counter*)
- CPSR: statusni register  
(*Current Program Status Register*)

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (SP)
r14 (LR)
r15 (PC)

CPSR
------

# Statusni register – CPSR

CPSR - Current  
Program Status  
Register



- zastavice (N,Z,V,C)
- maskirna bita za prekinitve (I, F)
- bit T določa nabor ukazov:
  - T=0 : ARM arhitektura, procesor izvaja 32-bitni ARM nabor ukazov
  - T=1: Thumb arhitektura, procesor izvaja 16-bitni Thumb nabor ukazov
- spodnjih 5 bitov določa način delovanja procesorja
- v uporabniškem (neprivilegiranem) načinu lahko CPSR beremo; ukazi lahko spreminjajo le zastavice.

## **Zastavice (lahko) ukazi spreminjajo glede na rezultat ALE:**

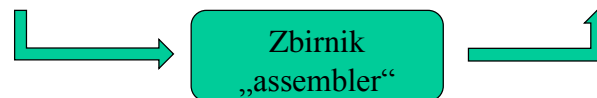
- N = 0: bit 31 rezultata je 0, N=1: bit 31 rezultata je 1 (Negative)  
Z = 1: rezultat je 0, Z=0: rezultat je različen od nič (Zero)  
C = 1: rezultat je povzročil prenos, C = 0: rezultat ni povzročil prenosa (Carry)  
V = 1: rezultat je povzročil preliv, V = 0: rezultat ni povzročil preliva (oVerflow)

# Programiranje v zbirniku

- **V zbirniku simbolično opisujemo:**

- ukaze (z mnemoniki),
- registre,
- naslove
- konstante

Zbirni jezik	Opis ukaza	Strojni jezik
ldr r1, stev1	<del><math>R1 \leftarrow M[0x20]</math></del>	0xE51F1014
ldr r2, stev2	<del><math>R2 \leftarrow M[0x24]</math></del>	0xE51F2014
add r3, r2, r1	<del><math>R3 \leftarrow R1 + R2</math></del>	0xE0823001
str r3, rez	<del><math>M[0x28] \leftarrow R3</math></del>	0xE50F3018



- **Programerju tako ni treba:**

- poznati strojnih ukazov in njihove tvorbe
- računati odmikov ter naslovov

**Prevajalnik za zbirnik (*assembler*) :**

- prevede simbolično predstavitev ukazov v ustrezne strojne ukaze,
- izračuna dejanske naslove ter
- ustvari pomnilniško sliko programa

- **Program v strojnem jeziku ni prenosljiv:**

- namenjen je izvajanju le na določeni vrsti mikroprocesorja

- **Zbirnik (*assembly language*) je „nizkonivojski“ programski jezik**



# Programiranje v zbirniku

- Vsaka vrstica programa v zbirniku predstavlja običajno en ukaz v strojnem jeziku
- Vrstica je sestavljena iz štirih stolpcev:

oznaka:	ukaz	operandi			@ komentar
↓	↓	↙	↓	↓	↓
rutina1:	add	r3	r3	#1	@ povečaj števec
	ldr	r5	[r0]		

- Stolpce ločimo s tabulatorji, dovoljeni so tudi presledki

# Ukazi

- **Vsi ukazi so 32-bitni**

```
add r3, r2, r1  $\implies$  0xE0823001=0b1110...0001
```

- **Vsi operandi se v registrih razširijo na 32-bitov**

```
0xFF  $\implies$  0x000000FF
```

- **Rezultat je 32-biten. Izjema je le množenje**

```
R1 + R2  $\implies$  R3
```

- **Aritmetično-logični ukazi so 3-operandni**

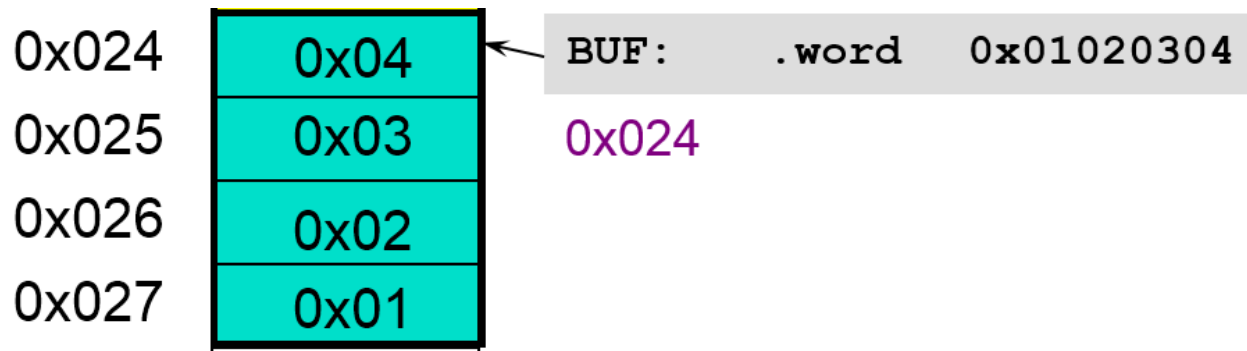
```
add r3, r3, #1
```

- **Load/store arhitektura**

```
ldr r1, stev1      @ prenos v registre  
ldr r2, stev2      @ prenos v registre  
add r3, r2, r1      @ vsota registrov
```

# Operandi

- 8, 16, 32-bitni predznačeni ali nepredznačeni pomnilniški operandi
- Obvezna poravnanost ukazov in operandov (16,32bitnih)
- V CPE se vse izvaja 32-bitno (razširitev ničle ali predznaka)
- Uporablja se pravilo tankega konca



# Oznake (labele)

Oznaka je nam razumljivo **simbolično poimenovanje** :

- pomnilniških lokacij ali
- vrstic v programu

Oznake običajno uporabljamo na dva načina:

- s poimenovanjem pomnilniških lokacij dobimo „spremenljivke“

```
STEV1:      .word   0x12345678
STEV2:      .byte   1,2,3,4
REZ:        .space  4
```

- za poimenovanje ukazov (vrstic), na katere se sklicujemo pri skokih.

```
      mov r4,#10
LOOP:  subs r4, r4, #1
      . . .
      bne LOOP
```

# Psevdoukazi - ukazi prevajalniku

## Psevdoukazi:

- so navodila prevajalniku
- običajno so označeni s piko pred ukazom
- niso strojni ukazi za CPE, temveč ukazi prevajalniku
- v končnem strojnem programu (izvaja CPE) jih ni

## Psevdoukaze uporabljamo za:

- določanje vrste pomnilniških odsekov **.text .data**
- poravnavo vsebine **.align**
- rezervacijo pomnilnika za „spremenljivke“ **.space**
- rezervacijo prostora v pomnilniku **.space**
- določanje začetne vsebine pomnilnika **.(h) word, .byte, ...**
- ustavljanje prevajanja **.end**

# Določanje pomnilniških odsekov

**Psevdoukaza za določanje pomnilniške slike sta:**

**.data**

**.text**

**S tema psevdoukazoma določimo, kje v pomnilniku bodo program(i) in kje podatki.**

**Tako za ukaze programa kot operande bomo uporabljali segment**

**.text**

# Rezervacija pomnilnika za „spremenljivke“

Za spremenljivke moramo v pomnilniku rezervirati določen prostor.

**RADIUS:**

Oznaka - ime  
„spremenljivke“

**.text**

**.align** @ **obvezna poravnost!**

**.space** 4 @ rezerviraj 4 bajte za RADIUS

Poravna na naslov deljiv s 4

Potrebujemo 4 bajte

**.align** @ **ukazi morajo biti poravnani!**

**ldr r7, RADIUS** @ v r7 nalozi RADIUS

Prevajalnik bo 'RADIUS' nadomestil z  
ustreznim naslovom lokacije – „spremenljivke“

# Rezervacija prostora v pomnilniku

Oznake omogočajo boljši pregled nad pomnilnikom:

– pomnilniškim lokacijam dajemo imena in ne uporabljamo absolutnih naslovov (preglednost programa)

<b>BUFFER:</b>	<b>.space 40</b>	<b>@rezerviraj 40 bajtov</b>
<b>BUFFER2:</b>	<b>.space 10</b>	<b>@rezerviraj 10 bajtov</b>
<b>BUFFER3:</b>	<b>.space 20</b>	<b>@rezerviraj 20 bajtov</b>

*;poravnano? Če so v rezerviranih blokih bajti, ni težav, sicer je (morda) potrebno uporabiti .align*

- oznaka **BUFFER** ustreza naslovu, od katerega naprej se rezervira 40B prostora
- oznaka **BUFFER2** ustreza naslovu, od katerega naprej se rezervira 10B prostora. Ta naslov ja za 40 večji kot **BUFFER**
- oznaka **BUFFER3** ustreza naslovu, od katerega naprej se rezervira 20B prostora. Ta naslov ja za 10 večji kot **BUFFER2**



# Rezervacija prostora z zač. vrednostmi

Večkrat želimo, da ima spremenljivka neko začetno vrednost.

```
niz1:    .asciz      "Dober dan"
niz2:    .ascii      "Lep dan"
         .align
stev1:    .word       512,1,65537,123456789
stev2:    .hword      1,512,65534
stev3:    .hword      0x7fe
Stev4:    .byte       1, 2, 3
         .align
naslov:   .word       niz1
```

- „spremenljivke“, inicializirane na ta način, lahko kasneje v programu spremenimo (ker so le naslovi pomnilniških lokacij)
- če želimo, da je oznaka vidna tudi v drugih datotekah projekta, uporabimo psevdoukaz `.global`, npr:

```
.global niz1, niz2
```

## Povzetek – psevdoukazi

