# Arhitektura računala 1- blicevi ARM

**4. blic – Ponavljanje gradiva**

1. Ako procesor ARM radi u načinu rada Fast Interrupt, kod pristupa registru R8 stvarno će se pristupiti fizičkom registru: R8_fiq.

2. Koja od sljedećih instrukcija pomiče sadržaj registra R0 aritmetički u desno za 8 mjesta i rezultat sprema natrag u registar R0: MOV R0, R0, ASR #8.

3. Na procesoru ARM izvodi se sljedeći programski odsječak: MOV r0, #8; MOV r1, #10; RSB r2, r0, r1. Nakon izvođenja instrukcija RSB sadržaj registra R2 će biti: 008.

4. Koja od sljedećih naredbi izvodi dohvat predznačnog okteta (bajta) u registar R0 sa memorijske lokacije na koju pokazuje registar R1: niti jedna od navedenih.

5. Koja od sljedećih naredbi neće izvesti skok u tijeku izvođenja programa: sve instrukcije su naredbe skoka (BL LABELA_1; MOV R15, R14; LDR R15, [R0]; ADDS R15, R0, R2).

6. Kod procesora ARM naredba MOV PC, #0: Označava skok na početak memorije (adresu 0).

7. Na procesoru ARM želimo pohraniti neposredne konstante u registre tako da koristimo naredbu MOV ili MVN (koja pohranjuje komplement neposredne vrijednosti). Koju od navedenih konstanti na ovaj način nije moguće pohraniti? Sve ih je moguće pohraniti ( 00004100, 0FFFFFFF, 0000BC00, 0AC00000).

8. Za procesor ARM napisan je sljedeći programski odsječak: ORG 0; MOV r0, pc; ADD r0, r0, #1. Nakon izvođenja tog odsječka, sadržaj registra R0 će biti: 9.

U PC-u arma se nalazi adresa koja glasi -> trenutna pozicija u programu + 8 -> zašto -> protočna struktura arma ima tri faze, pobliže ćemo učiti još što je u svakoj fazi radi, ali uvijek se dohvaća naredba koja je treća po redu u kodu od trenutne pozicije
-> trenutna pozicija na početku je bila 0 (zbog org 0), sadržaj PC-a je 8
-> sada je sadržaj r0 = 8
-> nakon naredbe add r0 = 8 +1 = 9

9. Na procesoru ARM naredba LDR R0, [R1], R2, LSL #2 radi sljedeće: Čita riječ sa memorijske lokacije R1, sprema ju u registar R0, te mijenja sadržaj registra R1=R1+4*R2.

10. Na procesoru ARM naredba STRH R1, [R2], -R3 radi sljedeće: Spremi poluriječ iz R1 na adresu R2 i nakon toga umanji registar R2 za vrijednost registra R3.

11. Neka je dan sadržaj registara R0=1, R1=2, R2=4, R3=2. Sadržaj registra R0 nakon izvođenja naredbe ADD R0, R1, R2, LSL R3 je: $12_{(16)}$.

12. Kod procesora ARM parametri se u potprogram ne mogu prenijeti preko: GPIO sklopa.

13. Na procesoru ARM izvodi se sljedeći programski odsječak:

```
        ...
        LDR   R0,  A
        LDMDB R0!, {R1,R2,R3}
        ...
A       DW  100
        ...
        DW  1 ; ovaj podatak je na adresi 0F4
        DW  2 ; ovaj podatak je na adresi 0F8
        DW  3 ; ovaj podatak je na adresi 0FC
        DW  4 ; ovaj podatak je na adresi 100
        DW  5 ; ovaj podatak je na adresi 104
        DW  6 ; ovaj podatak je na adresi 108
```

Nakon izvođenja instrukcije LDMDB sadržaji registara R0,R1,R2,R3 bit će redom:
a) 0F8, 1, 2, 3
b) 100, 2, 3, 4
**c) Ništa od navedenog (točan odgovor bio bi 0F4, 1, 2, 3)**
d) 100, 1, 2, 3
e) 0F4, 2, 3, 4

U r0 ide vrijednost 100 jer smo ucitali taj podatak s labele
-> load-amo vise stvari (LDM) i koristimo decrement before (DB)
-> imamo usklicnik iza r0 sta znaci da ce se on nakon naredbe promjeniti
-> posto imamo DB, r0 se prvo smanji za 4, i onda ucitavamo podatak (adresa 0FC) -> ide u r0
-> posto imamo DB, r0 se ponovno smanji za 4 i onda ucitavamo podatak (adresa 0F8) -> ide u r2
-> posto imamo DB, r0 se ponovno smanji za 4 i onda ucitavamo podatak (adresa 0F4) -> ide u r1
-> r0 ostaje na posljednjoj vrijednosti a to je 0F4
-> ovakav raspored ucitavanja u registre je zbog pravila kod multiple naredbi -> u najnize registre (s najmanjim brojem) idu vrijednosti na najmanjim lokacijama
-> tocan odgovor je C

Kod upotrebe naredbi sa nastavkom MULTIPLE to je jednostavno tako (ARM je takav :) )
Kod upotrebe naredbi sa nastavkom MULTIPLE moras racunat da ce se (ma kako bio postavljen nastavak npr. IB, IA, DB, DA) kada se ucitavaju podaci, podatak koji se nalazio na lokaciji sa najmanjom adresom ucitati u registara u viticastim zagradama koji ima namjanji 'broj',podatak na najvecoj lokaciji ide u registar s najvecim 'brojem', analogno za upisivanje u memoriju, podatak iz registra s najmanjim 'brojem' ce se prvi upisati, na kraju ce se podatak koji se nalazio u registru s najvecim 'brojem' upisati na posljednju zadanu lokacij

**4. blic – Pitanja s materijala**

1. Koja naredba pohranjuje najnižu poluriječ iz registra R3 u memorijsku lokaciju na koju pokazuje registar R2: STRH R3, [R2].

2. Početna heksadekadska vrijednost R13 je 4000. Nakon izvođenja naredbe LDMEA R13!, {R0} u R13 će biti heksadekadska vrijednost: 3FFC.

3. Podaci različitih širina kod ARMa se označavaju sa: B, H, W.

4. Sadržaj ARMovog registra R3=1AAAAA. Nakon izvođenja naredbe CLZ R1, R3 heksadekadski sadržaj registra R1 bit će: B.

5. Nakon izvođenja navedenog programskog odsječka na ARMu, heksadekadski sadržaji registara R0 i R1 će biti redom:

```
PRG   MOV  R0, #90
      LDMIB R0, {R1, R2}
      …
      ORG 90
      DW 102
      DW 103
      DW 104
```

Rješenje: 90 i 103.

RO = 90
R1 = 103
R2 = 104

R0 ostaje isti jer nema ! u naredbi LDMIB.
R1 = 103 jer LDMIB (IB = increment before). Prvo se uveca 90 + 4 = 94, te se vrijednost na adresi 94 ubaci u R1.
R2 = 104 : isto kao i za R1. Prvo se provede 94 + 4 = 98. I vrijednost s adrese 98 se upise u R2.

Napomena: da je naredba glasila ovako: LDMIB R0, {R2,R1}, rezulatat bi bio isti kao i gore. Jer ARM nize adrese pridruzuje "nizim" registirma, odnosno registirma s manjim indeksom.

6. Početne vrijednosti su R1=5 i R2=8. Nakon izvođenja ADD R0, R1, R2, LSR #1 vrijedi: $R0=9_{10}$.

7. Koja je od sljedećih tvrdnji točna za naredbu B kod procesora ARM: Naredba B može imati sufiks (polje uvjeta cond) kao uvjet za skok.

8. ARMova naredba STR, R0, [R1, R2, LSR #2] radi sljedeće: riječ iz R0 upisuje u memoriju na adresu R1+R2/4.

9. Početna heksadekadska vrijednost R13 je 7000. Nakon izvođenja naredbe LDMFD R13!, {R0} u R13 bit će heksadekadska vrijednost: 7004.

10. Za procesor ARM i naredbu BL POTP vrijedi tvrdnja: naredba BL upisuje adresu POTP u registar R15.

11. Uvjetno izvođenje naredbe kod ARMa ovisi o: Stanju zastavica u registru CPSR.

12. Koju od navedenih neposrednih heksadekadskih vrijednosti nije moguće pohraniti u registar korištenjem naredbe MOV: 201.

13. Programsko brojilo kod procesora ARM je: Registar R15.

14. Format naredbe LDR|STR {<cond>} {B} Rd, <addresing_mode> iz tablice objašnjava način pisanja nekoliko naredaba, a između njih i: Naredbe LD/ST za 32-bitne podatke.

15. Jedan od koraka pri izvođenju naredbe BL POTP je: U registar R15 stavlja se adresa POTP.

16. Kod izvođenja naredbe LDRSH viših 16 bita u registru bit će popunjeni: Bitom predznaka (najvišim bitom iz učitane poluriječi).

17. ARMova naredba STR R0, [R1], R2, LSL #2 radi sljedeće: Riječ iz R0 upisuje u memoriju na adresu R1, a zatim u R1 upisuje vrijednost R1+4*R2.

18. Podaci različitih širina kod ARMa se označavaju sa: B, H, W.

19. Registar CPSR dostupan je: U svim načinima rada.

20. Početne vrijednosti su R1=5, R2=7 i R3=1. Nakon izvođenja ADD R0, R1, R2, LSR R3 vrijedi: R0=$8_{10}$.

21. Početna heksadekadska vrijednost R13 je 8000. Nakon izvođenja naredbe STMFD R13!, {R0} u R13 bit će heksadekadska vrijednost: 7FFC.

22. Koja naredba pohranjuje najvišu poluriječ iz registra R10 u memorijsku lokaciju na koju pokazuje registar R6: nijedna od navedenih (STRB R10, [R6]; LDRB R10, [R6]; LDRHB R10, [R6]).

23. Prije izvođenja naredbe STMIB R13, {R0-R1} heksadekadski sadržaj registra R13 je 9000. Nakon izvođenja naredbe heksadekadski sadržaj R13 je: 9000.

24. Početne vrijednosti su R1=3 i R2=4. Nakon izvođenja ADD R0, R1, R2, LSL #1 vrijedi: R0=$11_{10}$.

25. Nakon izvođenja sljedećeg odsječka na ARMu, u memoriji na heksadekadskim adresama 90, 94, 98 i 9C bit će redom upisane heksadekadske vrijednosti:

```
PRG   MOV R0, #94
      STMIA R0, {R1, R2}
      …
      ORG 90
      DW 102
      DW 103
      DW 104
      DW 105
```

Rješenje: 102, R1, R2, 105.

26. Sadržaj ARMovog registra R3=1AAAAA. Nakon izvođenja naredbe CLZ R1, R3 heksadekadski sadržaj registra R1 bit će: B.

27. Ako na ARMu želimo skočiti na bilo koju 32-bitnu adresu možemo upotrijebiti: naredbu MOV R15, opci_registar.

28. ALU u procesoru ARM je: 32-bitna.

29. ARMova naredba STR R0, [R1, R2, LSR #3]! radi sljedeće: Riječ iz R0 upisuje u memoriju na adresu R1+R2/8, a zatim u R1 upisuje vrijednost R1+R2/8.

30. Povratak iz potprograma, čija adresa je PROC, na ARMu se ostvaruje ovako: Ništa od ponuđenog (U registar R15 stavlja se vrijednost iz registra R13; u registar R15 stavlja se adresa PROC; u registar R14 stavlja se podatak s vrha stoga).

31. Nakon izvođenja navedenog programskog odsječka na ARMu, heksadekadski sadržaji registara R0 i R1 bit će:

```
PRG   MOV R0, #68
      LDMDA R0, {R1, R2}
      …
      ORG 60
      DW 102
      DW 103
      DW 104
```

Rješenje: 68 i 103.

32. Koja od sljedećih naredaba dohvaća riječ u registar R0 iz memorijske lokacije na koju pokazuje registar R2 umanjen za 4: LDR R0, [R2, #-4].

33. U ARMovom registru R5 je vrijednost 55FFFF. Nakon izvođenja naredbe CLZ R2, R5 u registru R2 bit će dekadska vrijednost: 9.

34. Arhitektura procesora ARM je: Load-store.

35. Kod izvođenja naredbe STRSB viših 24 bita u memoriji bit će popunjeni: ova naredba ne postoji.

36. Registar CPSR dostupan je: u svim načinima rada.

37. Početne vrijednosti su R1=5, R2=7 i R3=1. Nakon izvođenja ADD R0, R1, R2, LSR R3 vrijedi: $R0=8_{10}$.

38. Gdje se sprema povratna adresa: U R14.

39. Za naredbu BL ADRESA vrijedi: ADRESA se upisuje u R15.

40. Početne vrijednosti registara su R1=5, R2=8. Nakon izvođenja naredbe ADD R0, R1, R2, LSR #1, koja je vrijednost u R0: R0=9 ( R2=8 pomakne se za jedno mjesto udesno i postane 4, R1=4+5=9).

41. Kod LDR i STR, odmak može biti: Registar opće namjene.

42. ARM je: RISC.

43. Prije naredbe STRIB R0, {R0-R1}, u R0 je 0x900. Nakon naredbe, R0 je:0x900 (nema !-->osvježenja)

44. Naredba STR R0, [R1, R2, LSL #3]: sprema R0 na adresu R1+R2*8.

45. U R1 upisana je konstanta 0x2FF. Nakon izvođenja naredbe CLZ R4, R1 u R4 će biti upisano: $22_{16}$ (0x2FF=**0000 0000 0000 0000 00**10 1111 1111)

46.

```
      ORG 0
PRG   MOV R0, #90
      LDRIB R0, {R1, R2}
      …
      ORG 90
      DW 0x100
      DW 0x101
      DW 0x102
```

Kolika je vrijednost registara R0 i R1: R0 je sigurno 0x90, a R1=0x100.

47. Početne vrijednosti su R1=5 i R2=8. Nakon izvođenja ADD R0, R1, R2, LSR #1 vrijedi: R0=$9_{10}$.

**4. blic – Pitanja s FER2.net**

1. Za spremanje povratne adrese iz potprograma/iznimke kod ARMa se koristi: Registar R14.

2. Za sve naredbe load/store se adresa memorije izračunava korištenjem: Baznog registra i odmaka.

3. U ARMovom registru R5 je vrijednost 55FFFF. Nakon izvođenja naredbe CLZ R2, R5 u registru R2 bit će dekadska vrijednost: (55FFFF=**0000 0000 0**101 0101 1111 1111 1111 1111) 9.

4. Koju od navedenih neposrednih heksadekadskih vrijednosti nije moguće pohraniti u registar korištenjem naredbe MOV: 4020

5. Za naredbu BL ADRESA na procesoru ARM vrijedi tvrdnja: naredba BL upisuje ADRESA u registar R15.

6. Nakon što ARM izvede sljedeći programski odsječak, heksadekadski sadržaji registara R0, R1 i R2 bit će redom:

```
PRG   MOV R0, #98
      LDMDB R0, {R1, R2}
      …
      ORG 90
      DW 102
      DW 103
      DW 104
```

Rješenje: 98, 102, 103

7. Početne vrijednosti su R1=2, R2=4, R3=2. Nakon izvođenja ADD R0, R1, R2, LSL R3 vrijedi: R0=1810.

8. Početna heksadekadska vrijednost R13 je 9000. Nakon izvođenja naredbe STMED R13!, {R0} u R13 bit će heksadekadska vrijednost: 8FFC.

9. Jednoznačno definirani registri opće namjene u procesoru ARM su: registri R0-R7.

10. ARMova naredba STR R0, [R1, R2, LSL #3] radi sljedeće: Riječ iz R0 upisuje u memoriju na adresu R1+8*R2.

11. Koja od sljedećih naredaba dohvaća riječ u registar R0 iz memorijske lokacije na koju pokazuje registar R2 umanjen za 4: LDR R0, [R2, #-4].

12. Sve osnovne operacije ARMa obavljaju se nad: 32-bitnim podacima.

13. Nakon izvođenja navedenog programskog odsječka na ARMu, heksadekadski sadržaji registara R0 i R1 bit će redom:

```
PRG   MOV R0, #70
      LDMIA R0, {R1, R2}
      …
      ORG 70
      DW 102
      DW 103
      DW 104
```

Rješenje: 70 i 102.

14. Programsko brojilo kod procesora ARM je: registar R15.

15. ARMova naredba STR R0, [R1, R2, LSL #3] radi sljedeće: Riječ iz R0 upisuje u memoriju na adresu R1+8*R2.

16. Koja naredba pohranjuje najniži bit iz registra R0 u memorijsku lokaciju na koju pokazuje registar R5: STRB R0, [R5].

17. Podaci različitih širina kod ARMa označavaju se sa: B, H, W.

18. Početne vrijednosti su R1=3, R2=4. Nakon izvođenja ADD R0, R1, R2, LSL #1 vrijedi: $R0=11_{10}$.

19. U ARMovom registru R5 je vrijednost 55FFFF. Nakon izvođenja naredbe CLZ R2, R5 u registru R2 bit će dekadska vrijednost: 9.

20. Kod izvođenja naredbe LDRB viših 24 bita u registru bit će popunjeni: nulama.

21. Za procesor ARM i naredbu BL POTP vrijedi tvrdnja: Naredba BL upisuje adresu POTP u reg. R15.

22. Arhitektura procesora ARM je: Load-store.

23. ARMova naredba STR R0, [R1, R2, LSL #2] radi sljedeće: Riječ iz R0 upisuje u memoriju na adresu R1+4*R2.

24. Početna heksadekadska vrijednost R13 je 9000. Nakon izvođenja naredbe LDMED R13!, {R0} u R13 bit će heksadekadska vrijednost: 9004.

25. Koju od navedenih neposrednih heksadekadskih vrijednosti nije moguće pohraniti u registar korištenjem naredbe MOV: 4040.

Definirati neposrednu vrijednost možeš samo ako se taj broj može dobiti rotacijom 8-bitnog broja za paran broj mjesta u desno.
tj. 4040 je **100000001**000000 i to ne možeš dobiti rotacijom 8-bitnog broja za paran broj mjesta, dok broj 2080 je **1000001**0000000 i to možeš dobiti rotacijom 8-bitnog broja.

26. Jednoznačno definirani registri opće namjene u procesoru ARM su: registri R0-R7.

27. Kod izvođenja naredbe LDRH viših 16 bita u registru bit će popunjeni: nulama.

28. Kod ugniježđenih poziva potprograma na ARMu, u potprogramima treba: Pohraniti povratnu adresu potprograma na stog kako bi je kasnije mogli dohvatiti.

**4. blic 2013-2014**

1. Početne vrijednosti su R1=5 i R2=8. Nakon izvođenja ADD R0, R1, R2, LSR #1 vrijedi: $R0=9_{10}$.

2. Koja od sljedećih naredaba sprema sadržaj registra R5 u memorijsku lokaciju na koju pokazuje registar R1 uvećan za 4: STR R5, [R1, #4].

3. Sadržaj ARMovog registra R3=0x1AAAAA. Nakon izvođenja naredbe CLZ R1, R3 sadržaj registra R1 bit će: 0XB. (R3 u binarnom obliku: 0000|0000|0001|1010|1010|1010|1010|1010 (obavezno gledati sadržaj cijelog 32-bitnog registra)

4. Početna vrijednost R13 je 0x5000. Nakon izvođenja naredbe STMEA R13!, {R0} u R13 bit će: 0x5004 (budući da se iza adresnog registra (R13) nalazi uskličnik, njegova vrijednost će se osvježiti. STMEA znači Store Multiple Empty Ascend (gdje Empty Ascend možete zamisliti kao da je sadržaj adrese na koju pokazuje R13 prazna (odatle Empty) pa će to biti prva adresa na koju se sadržaj R0 sprema, a Ascend znači da se vrijednost povisuje za 4 bajta).

5. Naredbe se kod ARMa uvijek nalaze na adresi: koja ima dva najniža bita 00 (s predavanja: "Valja uočiti da su pri dohvatu naredbe najniža dva bita uvijek postavljena u logičku nulu zbog toga jer su naredbe ARM-a uvijek poravnate na širinu rijeci." - prezentacija 19, slajd 20).

6. ARMova naredba STR R0, [R1, R2, LSL #2] radi sljedeće: riječ iz R0 upisuje u memoriju na adresu R1+4*R2 (iza adrese nema uskličnika, što znači da R1 ostaje nepromijenjen. Jedino kod naredbe STR prvi operand nije destinacijski, nego izvorni, stoga se sadržaj iz R0 sprema na adresu R1 + (R2 koji je logički rotiran ulijevo za 2 bita).

7. Nakon što ARM izvede sljedeći programski odsječak, sadržaji registara R0, R1 i R2 bit će redom:

```
PRG     MOV R0, #50
        LDMIB R0, {R2, R1}
        …
        ORG 50
        DW 102
        DW 103
        DW 104
```

Rješenje: 0x50, 0x103, 0x104 (Load Multiple Increment Before - ovdje treba paziti da bez obzira na redoslijed registara u vitičastim zagradama, uvijek će se vrijednosti iz memorije učitavati s nižih adresa u registre s nižim brojem. Budući da iza R0 nema uskličnika, njegova vrijednost ostaje na #50. Zbog sufiksa IB, adresa se poveća s 50 na 54 pa se tek onda učita saržaj s adrese 54 (a to je hex 103) u registar R1, itd.)

8. Za naredbe LDR/STR odmak u adresi može biti: Neposredna vrijednost (isključite sve netočne odgovore i ostat će jedino D (primjer s neposrednom vrijednošću se nalazi u samom blicu u 2. zadatku).

9. Programsko brojilo kod procesora ARM je: registar R15 (PC je posljednji registar kojemu mi kao programeri možemo pristupiti, R15).

10. Za naredbu BL ADRESA na procesoru ARM vrijedi tvrdnja: Naredba BL sprema povratnu adresu u R14. (registar R14 je definiran kao Link Register, u kojega naredba BL sprema povratnu adresu).

11. Za procesor ARM točna je tvrdnja: Naredbe ARMa se mogu izvoditi uvjetno. (isključite sve netočne odgovore i ostat će jedino D (također ovo su valjda 20 puta spomenuli na predavanjima).

12. Prije izvođenja naredbe LDMIB R13, {R5, R0-R1} sadržaj registra R13 je 0x6000. Nakon izvođenja naredbe sadržaj R13 je: 0x6000 (budući da iza R13 nema uskličnika, njegov se sadržaj će ostati nepromijenjen poslije naredbe za višestruko učitavanje).

13. Koja od sljedećih tvrdnji je točna za naredbu BL kod procesora ARM: Naredba BL može imati sufiks (polje uvjeta cond) kao uvjet za skok. (ovdje može nastati dvojba oko točnih i netočnih odgovora, ali ako pogledate u ARM7 šalabahter, oblik naredbe BL izgleda ovako: **BL**{cond} label, gdje je {cond} prostor za uvjet).

14. Nakon što ARM izvede sljedeći programski odsječak, sadržaji registara R0, R1, R2 bit će redom:

```
PRG   MOV R0, #98
      LDMDB R0, {R1, R2}
      …
      ORG 90
      DW 102
      DW 103
      DW 104
```

Rješenje: 0x98, 0x102, 0x103. (budući da iza R0 nema uskličnika, on na kraju ostaje kakav je bio. LDMDB znači Load Multiple Decrement Before, stoga se adresa sa 98 prvo smanji na 94 pa se učita njen sadržaj u **registar s većim indeksom - R2**. Zatim se smanji na 90 pa se sadržaj s te lokacije (102) učita u prvi niži registar - R1. Bitno je znati da je ARM konstruiran tako da se kod naredbi za višestruko učitavanje i spremanje uvijek učitavaju i spremaju podaci s **nižih adresa** u registre s **nižim indeksom** i sve prema većemu).

15. Format naredbe LDR|STR {<cond>} {B} Rd, <addressing_mode> iz tablice objašnjava način pisanja nekoliko naredaba, a između njih i: Naredbe LD/ST za 32-bitne podatke.

16. Kod procesora ARM direktan upis vrijednosti u registar R7 uzrokovat će: Neće uzrokovati ništa jer je to registar opće namjene. (registar R7 je jednoznačno definirani registar opće namjene).

17. ARMova naredba STR R0, [R1, R2, LSR #3]! radi sljedeće: Riječ iz R0 upisuje u memoriju na adresu R1+R2/8; zatim u R1 upisuje vrijednost R1+R2/8. (sadržaj registra R0 se sprema na adresu [R1 + (R2 koji je logički pomaknut udesno za 3 bita == dijeljenje s 8)]. Budući da se iza adrese nalazi uskličnik, u R1 će se upisati ista ta adresa u koju je upisan i sam podatak R0).

18. Koja od sljedećih naredaba sprema sadržaj registra R2 u memorijsku lokaciju na koju pokazuje registar R11 uvećan za sadržaj registra R4: STR R2, [R11, R4]. (čitajući ovaj zadatak, najbolje si je odmah napisati što traže, i sigurno ćete dobiti jedan od ponuđenih rješenja. Pritom naravno treba paziti na sintaksu u ARM-u).

19. U ARMovom registru R5 je vrijednost 0xFF0F. Nakon izvođenja naredbe CLZ R0, R5 u registru R0 bit će vrijednost: 0x10. (R5 u binarnom obliku: 0000|0000|0000|0000|1111|1111|0000|1111 (obavezno gledati sadržaj cijelog 32-bitnog registra). Također, treba se paziti na heksadekadsku bazu).

20. Koju od navedenih neposrednih vrijednosti nije moguće pohraniti u registar korištenjem naredbe MOV: 0x4040 (binarni oblik hex broja 4040 je 0100|0000|0100|0000, a razmak između jedinica je preveliki pa se ne može dobiti neposrednom vrijednošću i rotacijom)

1) pretvoriš broj koji trebaš dobiti u binarni...
2) pogledaš dali sve jedinice stanu u 8 bitova (recimo, imaš broj 0001111011001, taj ne stane jer mu je razmak od jedinice skroz lijevo do one skroz desno 10 mjesta, dok kod 011100110 može jer je razmak 7)
3) ako ne stanu, broj se ne može zapisati, ako stanu, idemo dalje ;)
4) staviš taj 8-bitni dio skroz desno (u 32-bitnom broju u kojem je ostatak popunjen nulama) i gledaš za koliko mjesta treba rotrati taj 32 bitni broj da bi se dobio traženi broj. ako je broj mjesta paran, onda se može, ako nije, onda se ne može zapisati.
5) ako se može zapisati, onda je broj rotacija / 2 = R. recimo da je trebalo 20 rotacija, R=10, odnosno 1010... onda bi konstanta koju treba upisati bila: 1010<onih 8 bitova>, recimo 101011100101.
6) ako možeš koristiti i MVN, onda za one konstante koje nisi pomoću MOV mogo napisati napravi komplement i probaj ponovo.)