1) Računalnik z navideznim pomnilnikom z ostranjevanjem ima čas dostopa do glavnega pomnilnika 55 ns. Čas za prenos strani iz navideznega pomnilnika v glavni pomnilnik je 9 ms, verjetnost za napako strani pa je 10^{-6}. Kakšen je povprečni dostopni čas če:

a) sta tabeli strani v glavnem pomnilniku,
b) tabeli strani sta v glavnem pomnilniku, imamo pa preslikovalni predpomnilnik z verjetnostjo zadetka 98%.

V obeh primerih je preslikava navideznega naslova v fizičnega dvonivojska (preko dveh tabel strani).

Rešitev:

a)
$t_a = t_{ag} + t_{ag} + t_{ag}$+1E-6*9ms
$t_a = 55ns + 55ns$ + 55ns + 9ns
t_a = 174ns

b)
$t_a = 0.02*[t_{ag}+t_{ag}]$+ tag+1E-6*9ms
$t_a = 0.02*[55ns+55ns]$ + 55ns + 9ns
t_a = 66.2ns

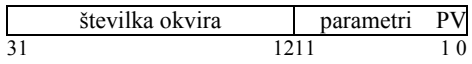
2) Na računalniku z navideznim pomnilnikom poženemo program za kopiranje pomnilniškega bloka velikosti 64B od naslova 0x00000FD0 dalje na naslove od 0x00001FE0 dalje. Pomnilniška beseda je dolga 1B, pri kopiranju se uporabljajo 32-bitni dostopi. Dolžina navideznega in fizičnega naslova je 32 bitov, velikost strani 4KB, preslikovanje naslovov je enonivojsko. Dostopni čas do glavnega pomnilnka je 55ns, čas za prenos strani iz navideznega v glavni pomnilnik znaša 9ms. Tabela strani je v glavnem pomnilniku od naslova 0xFFF00000 dalje; del tabele strani in zgradba deskriptorja strani sta prikazani na sliki.

a) V katera fizična naslova se preslikata navidezna naslova 0x00000FD0 in 0x00001FE0?

b) Izračunajte povprečni dostopni čas pomnilniškega dostopa do operanda pri zgoraj navedenem kopiranju 64B dolgega bloka (bralni dostopi od naslova 0x00000FD0 dalje in pisalni dostopi od 0x00001FE0 dalje)? Kolikšen bi bil ta čas v primeru, če pri izvajanju programa ni nobene napake strani.

c) V računalnik vgradimo TLB (čisti asociativni) velikosti 32 deskriptorjev strani, ki pred izvajanjem programa ne vsebuje nobenega deskriptorja strani z operandi. Kakšen je odgovor na vprašanji iz točke b) v tem primeru?

Deskriptor in tabela strani:



V – bit 0 (stran prisotna): 1-da / 0-ne
P – bit 1 (veljavni bit): 1-veljavno / 0-neveljavno

naslov	vsebina
0xFFF00000	0x00000003
0xFFF00004	0x00007003
0xFFF00008	0x00005001
0xFFF0000C	…
…	…

Rešitev:

a) 0x00000FD0 → 0x00000FD0, 0x00001FE0 → 0x00007FE0

b) dostopi:

	branje	pisanje
1	0x00000FD0,	0x000001FE0
2	0x00000FD4,	0x000001FE4
3	0x00000FD8,	0x000001FE8
4	0x00000FDC,	0x000001FEC
5	0x00000FE0,	0x000001FF0
6	0x00000FE4,	0x000001FF4
7	0x00000FE8,	0x000001FF8
8	0x00000FEC,	0x000001FFC
9	0x00000FF0,	**0x000002000**
10	0x00000FF4,	0x000002004
11	0x00000FF8,	0x000002008
12	0x00000FFC,	0x00000200C
13	0x00001000,	0x000002010
14	0x00001004,	0x000002014
15	0x00001008,	0x000002018
16	0x0000100C,	0x00000201C

Pri poudarjenem pride do napake strani (to se zgodi samo enkrat). Torej imamo 31 dostopov, ko napake strani ni, pri enem pa imamo napako strani. Predpostavimo, da ob napaki strani prenos strani iz okvira v glavnem pomnilniku v navidezni pomnilnik ni potreben:

t_{acc}=[31*(55ns+55ns) + **55ns+55ns+9ms**] / 32 = 281,3us

če napake strani ne bi bilo, imamo 32 dostopov brez napake strani:

t_{acc}=[32*(55ns+55ns)] / 32 =110ns

c) dostopamo do treh strani (v TLB se iz tabele strani prenesejo trije deskriptorji):

t_{acc}= [
3 * 55ns (tabela strani)

+32 * 55ns (dostop v fiz.pom.)
+1*9ms (napaka strani)]/32
= 28,13us

če napake strani ne bi bilo:

t_{acc}= [
3 * 55ns (tabela strani)
+32 * 55ns (dostop v fiz.pom.)]/32
= 60,15ns

3) Na računalniku z navideznim pomnilnikom na osnovi ostranjevanja poženemo program dolžine 48 B, ki se nahaja na naslovih od 0x000401FE0 dalje. Program z 32-bitnimi dostopi prekopira pomnilniški blok velikosti 1,048576 MB od naslova 0x000100000 dalje na naslove od 0x000300000 dalje. Dolžina navideznega naslova je 36 bitov, dolžina fizičnega naslova je 32 bitov, velikost strani je 4,096 KB, pomnilniška beseda je dolga 1 B, preslikovanje naslovov je dvonivojsko, pri čemer je velikost tabele strani 2. nivoja 65536 B. Vse tabele strani so v glavnem pomnilniku.

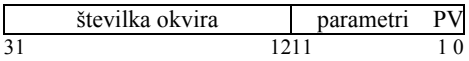
Na sliki je prikazana vsebina tabele strani 2. nivoja in ene izmed tabel strani 1. nivoja pred izvajanjem programa in format deskriptorja strani, ki velja za tabele strani obeh nivojev. Vse ostale tabele strani 1. nivoja so prazne (v njih so zapisane ničle).

a) Kakšna je velikost tabele strani 1. nivoja v bajtih? Utemeljite.

b) Na katerih fizičnih naslovih se nahaja program? Utemeljite.

c) Kakšen je povprečni dostopni čas do podatkov pri kopiranju (upoštevajte le bralne in pisalne dostope pri kopiranju podatkov), če je dostopni čas do glavnega pomnilnika 50 ns. Predpostavite, da je TLB na začetku prazen, da se pri izvajanju programa v TLB pojavljajo le obvezne zgrešitve in zgrešitvena kazen ob napakah strani znaša 10 ms.

Format deskriptorja strani:



V – bit 0 (stran prisotna): 1-da / 0-ne
P – bit 1 (veljavni bit): 1-veljavno / 0-neveljavno

Tabela strani 2. nivoja:

naslov	vsebina
0xFFF00000	0xFFF01003
0xFFF00004	0xFFF03003
0xFFF00008	0xFFF04003
0xFFF0000C	0xFFF05003
0xFFF00010	0xFFF02003
0xFFF00014	0x00000000
…	…

Ena izmed tabel strani 1. nivoja:

naslov	vsebina
0xFFF03000	0x000A0003
0xFFF03004	0x000A2003
0xFFF03008	0x000A1003
0xFFF0300C	0x00000000
0xFFF03010	0x00000000
0xFFF03014	0x00000000
…	…

Rešitev:

a)

ker je tabela strani na nivoju 2 velika 65536B in je deskriptor dolg 4B, je v njej 2^14 deskriptorjev in je številka deskriptorja na nivoju 2 v navideznem naslovu določena z biti 22-35.

Ker je stran dolga 4,096 KB in je dolžina pomnilniške besede 1B, potrebujemo za naslov v strani 12 bitov (vsak B ima svoj naslov, torej 2^12*1B=4,096KB)

Ostalih 10 bitov v navideznem naslovu (to so biti 12-21) torej določa deskriptor v tabeli strani 1. nivoja in ker je dolžina deskriptorja 4B, to pomeni, da je velikost tabel strani na 1. nivoju **2^10*4B=4,096KB**.

b)
program se nahaja na navideznih naslovih od 0x000401fe0 dalje in je dolg 48B, torej se nahaja na navideznih naslovih:

začetek: 0x000401fe0
konec: 0x000401fe0+48-1=0x00040200F

naslova pretvorimo v dvojiški sistem:

0x000401fe0 = 00000000000001000000000111111100000

V tabeli 2. nivoja se preslika preko deskriptorja 00000000000001

0xFFF03003 (podčrtana je številka okvira, v katerem je tabela strani nivoja 1. To je stran od naslova 0xFFF03000 dalje.)

V njej se naslov preslika preko deskriptorja 0000000001. Preveriti je potrebno, da sta bita P in V oba 1.

0x000A2003 (podčrtana je številka okvira, v kateri je stran). Preveriti je potrebno, da sta bita P in V oba 1.

Združimo številko okvira in naslov znotraj strani in dobimo fizični naslov:

0x0000A2FE0

0x000402007 = 00000000000001000000001000000000007

V tabeli 2. nivoja se preslika preko deskriptorja 00000000000001

0xFFF03003 (podčrtana je številka okvira, v katerem je tabela strani nivoja 1. To je stran od naslova 0xFFF03000 dalje.)

V njej se naslov preslika preko deskriptorja 0000000010. Preveriti je potrebno, da sta bita P in V oba 1

0x00A1003 (podčrtana je številka okvira, v kateri je stran). Preveriti je potrebno, da sta bita P in V oba 1

Združimo številko okvira in naslov znotraj strani in dobimo fizični naslov:

0x000A100F

Program se torej nahaja na fizičnih naslovih 0x000A2FE0 - 0x000A2FFF in 0x000A100 - 0x000A100F

c)

Kopira se 1 MB velik pomnilniški blok z naslovov od 0x000100000 na naslove od 0x000300000 dalje. Ker se kopira po 4 B naenkrat, je število dostopov 2 x 2^20/4 = 2^19. Število uporabljenih strani je 2 x 2^20B/2^12B = 2^9.

Ob prvem dostopu do vsake strani imamo:

- zgrešitev v TLB, za preslikavo naslova sta potrebna dva dostopa do glavnega pomnilnika: 2 x $2t_{ag}$
- zaradi napake strani znaša dostopni čas do podatka: t_{ag} + 10 ms.

= skupaj 3x t_{ag}+10 ms.

Ob nadaljnjih dostopih do iste strani:

- zadetek v TLB, ker imamo v TLB le obvezne zgrešitve: ni izgube časa
- ni napake strani, torej je dostopni čas do podatkov: t_{ag}

= skupaj t_{ag}

Povprečni dostopni čas za dostope do podatkov torej znaša:

$$t_a = \frac{512 * (3t_{ag} + 10ms) + (2^{19} - 512)t_{ag}}{2^{19}} = 9{,}8\mu s$$

4) Napišite podprogram za AT91SAM9260 (ARM) za pretvarjanje 16-bitnih števil iz zapisa v dvojiškem komplementu v zapis v eniškem komplementu. Podprogram sprejme tri parametre: v registru r0 naslov vhodnega polja 16-bitnih števil v dvojiškem komplementu, v r1 naslov izhodnega polja 16-bitnih števil v eniškem komplementu in v r2 število elementov v poljih. Navodilo: če je število pozitivno, ga je potrebno samo prekopirati iz vhodnega v izhodno polje. Če je negativno, mu je potrebno odšteti ena in rezultat zapisati v izhodno polje. Zagotovite, da so vrednosti registrov, ki jih poleg r0,r1 in r2 uporabljate v podprogramu, po izhodu iz podprograma nespremenjene. Sklad se širi proti nižjim naslovom, kazalec na sklad pa kaže na **zadnji element** na skladu. Pri ocenjevanju se upoštevajo pravilni načini naslavljanja in čim manjše (optimalno) število ukazov!

Rešitev:

```
        stmfd r13!, {r3, r14}
lp:     ldrsh r3, [r0], #2
        cmp r3, #0
        strhpl r3, [r1], #2
        submi r3, r3, #1
        strhmi r3, [r1], #2
        subs r2, r2, #1
        bne lp
        ldmfd r13!, {r3, pc}
```

Napišite podprogram v zbirnem jeziku za ARM procesor AT91SAM9260, ki naj preko DBGU enote pošlje blok 8-bitnih podatkov. Podprogram dobi v registru R0 naslov bloka podatkov, v registru R1 pa njegovo dolžino. Podprogram naj najprej pošlje dolžino bloka (spodnjih 16 bitov registra R1), nato podatke v bloku in na koncu še vsoto vseh podatkov v bloku po modulu 256, ki jo računate sproti med pošiljanjem posameznih podatkov bloka. Za pošiljanje enega 8-bitnega podatka že obstaja podprogram DBGU_SND, ki kot parameter v registru R0 prejme znak, ki naj se trenutno pošlje in ne spreminja vrednosti nobenega registra razen R0. DBGU enota je že ustrezno nastavljena. Upoštevajte, da mora podprogram ohraniti vrednost vseh registrov, ki jih uporablja razen R0 in R1.

Rešitev:

```
   stmfd sp!,{r2-r3 ,lr}

   mov r2, r0
   mov r0, r1
   and r0, r0, #0xFF
   bl  DBGU_SND
   mov r0,r1,lsr #8
   bl  DBGU_SND

   mov r3,#0
loop: ldrb r0,[r2],#1
   add r3,r3,r0
   bl DBGU_SND
   subs r1,r1,#1
   bne loop

   and r3,#0xFF
   mov r0,r3
   bl DBGU_SND

   ldmfd sp!,{r2-r 3,pc}
```