3. zadatak ZI 2011/2012

Postovani,

u proslogodisnjem ZI-ju, trecem zadatku (performanse), nije mi jasno da li se ucestalost promasaja odnosi na sve instrukcije ili samo na one koje dodu do te memorije (u¶ cestalost proma¶ saja L1, L2: 3%, 0.4% - da li je tih 0,4% od ukupnog broja ili je to samo od onih koji dodu do te memorije)?

Tj. koja je od ovih formula tocna:

1.
T1=N*1T+0.03*(N*10T+0.004*N*100T)
T2=N*100T <--- preskacemo L1 i L2 jer je doslo do zastoja prirucne memorije
Pogorsanje je T2/T1=76 puta.

2.
Tidealno = 1T
Trealno = 0.97 * 1T + 0.03 * 10T + 0.004 * 100T = 1.67

Lp,
SP

Njegov odgovor:

Spoiler:
Odgovor pod 1. ima vise smisla: kada se dogodi promasaj L1 pristupa se prvo L1 a potom i L2 pa je ukupno vrijeme zbroj navedenih vremena pristupa.

Rekao bih da je bolje racunati omjer T1/T2 jer performansu usporedujemo sa T2 (koje je uvijek najmanje).
Takoder nije 76 puta, vec 76 posto.

Lp,
BS

Pojašnjenje zadatka na 32. slide-u 7. Predavanja

w(p) - broj bitova ili širina za pomak

w(i) - broj bitova ili širina za indeks linije

w(o) - broj bitova ili širina za oznaku (značajku)

Računamo ih na sljedeći način:

w(p) = log2(b) gdje je b veličina linije ili w(p) = log2(b/zrnatost) (ako je u tekstu zadatka posebno naglašena zrnatost)

w(i) = log2(n) (ako imamo izravno preslikavanje) ili w(i) = log2(n/a) ako je u tekstu zadatka zadana asocijativnost a (to je kod asocijativnog preslikvanja)

w(o) = veličina adrese - w(p) - w(i)

Napomena: log2(x) - logaritam po bazi 2 broja x

E sad kad to imamo možemo preći na pogodak, promašaj i promašaj sa promjenom

1) Pogodak - preklapanje bitova indeksa (koji određuje liniju) i bitova oznake (značajke)

2) Promašaj - različiti bitovi indeksa i bitovi oznake

3) Promašaj s promjenom - preklapanje bitova indeksa, ali različiti bitovi oznake

Za gore navedeni primjer imamo:
a) 0x00000014 : linija 1, pomak 4 (promašaj) - uvijek promašaj na početku jer je mem. prazna (osim ako nije zadano drugačije) na liniju 1 pišemo taj podatak
b) 0x0000001C : linija 1, pomak C (pogodak) - na liniji jedan već imamo podatak iz a), radimo provjeru bitovi indeksa su 1, bitovi oznake su 0 imamo pogodak
c) 0x00000034 : linija 3, pomak 4 (promašaj) - na liniji 3 nemamo podatak, pa je promašaj, upisujemo taj podataka
d) 0x00008014 : linija 1, pomak 4 (promašaj s promjenom) - na liniji jedan imamo podatak, radimo provjeru bitovi indeksa su 1, no bitovi oznake su različiti imamo promašaj s promjenom, upisujemo taj podatak (radi se o izravnom preslikavanju)
e) 0x00000030: linija 3, pomak 0 (pogodak) - na liniji tri već imamo podatak iz c), radimo provjeru bitovi indeksa su 3, bitovi oznake su 0 imamo pogodak
f) 0x0000001C: linija 1, pomak c (promašaj s promjenom) - na liniji jedan imamo podatak iz d), radimo provjeru bitovi indeksa su 1, no bitovi oznake su različiti imamo promašaj s promjenom, upisujemo taj podatak (radi se o izravnom preslikavanju)
g) 0x00008020: linija 2, pomak 0 (promašaj) - na liniji 2 dosad nismo imali podataka, što je promašaj (mem. prazna), upisujemo taj podatak

Random zadatak nekakvo rješenje

Ovako:
a=2
s = 512 B
b = 16 B

zrnatost 1B, adrese 32-bitne
---
broj linija n = s/b = 32

w(i) = log(2) n/a = 4 bita
w(p) = log(2) b = 4 bita
w(o) = 32-4-4 = 24 bita

Znaci ovako nam to izgleda: oooooooooooooooooooooooiiiipppp

Obzirom da se radi o dvoasocijativnoj PM, imamo na raspolaganju 16 linija, a svaka ima 2 "kutije" (to si ja tak slikovito predocim).

Sad imamo:
0x12 -> oznaka 0, linija 1, pomak 2 -> PM je bila prazna, promasaj
0x13 -> oznaka 0, linija 1, pomak 3 -> to smo ucitali u prethodnom koraku, HIT
0x28 -> oznaka 0, linija 2, pomak 8 -> promasaj
0x514 -> oznaka 5, linija 1, pomak 4 -> promasaj, ali to spremamo u drugu kutiju na prvoj liniji
0x217 -> oznaka 2, linija 1, pomak 7 -> promasaj, algoritam zamjene LRU, stavljamo ovo u prvu kutiju na prvoj liniji

0x122 -> oznaka 1, linija 2, pomak 2 -> promasaj, ide u drugu kutiju na drugoj liniji
0x19 -> oznaka 0, linija 1, pomak 9 -> promasaj, na prvoj liniji trenutno imamo 0x514 i 0x217, stavljamo ovo umjesto 0x514, znaci druga kutija
0x512-> oznaka 5, linija 1, pomak 2 -> promasaj, jbg, taman smo to izbacili. stavljamo u prvu kutiju na prvoj liniji
0x11 -> pomak 0, linija 1, pomak 1 -> HIT, imamo to u drugoj kutiji na prvoj liniji.

Obzirom da nisam 100% siguran, i da sam malo na brzinu ovo radio, molim da netko potvrdi :)

http://materijali.fer2.net/File.7951.aspx

1.(3 boda) Kapacitet primarne memorije je 1M bajt, kapacitet sekundarne memorije je 1G bajt. Virtualni memorijski sustav temelji se na straničenju. Veličina stranice je 1K bajtova. Virtualni memorijski sustav, umjesto potpuno asocijativnog načina preslikavanja, koristi izravni (direktan) način preslikavanja. Izračunajte indeks straničnog priključka u koji će se moći priključiti stranica (iz sekundarne memorije) ako je adresa referenciranog podatke 4098 (dekadno!). Pozor: Zadatak se priznaje i donosi 3 boda samo ako se točno **izračuna** indeks straničnog priključka (i izrazi dekadno).

primarna 1 MB
sekundarna 1 GB
stranica 1 KB

n=primarna/veličina stranice = 1024 ->broj stranica

j = 4098/1024(velicina stranice) = 4 ("najveće cijelo") ->adresa **bloka u sekundarnoj memoriji**

i=j mod n = 4 mod 1024(broj stranica) = 4 -> indeks **stranice u glavnoj memoriji**

Hrkačev mail:

Dakle da rezimiramo, kod izravnog preslikavanja:
- stranica 0 iz virtualne memorije preslikava se na stranicu 0 u glavnoj memoriji,
- stranica 1 iz virtualne memorije preslikava se na stranicu 1 u glavnoj memoriji,
... (itd) ...
- stranica M-1 iz virtualne (gdje je M ukupan broj stranica u glavnoj mem; u nasem slucaju M = (1MB/1KB) = 1024) preslikava se na stranicu M-1 u glavnoj
- stranica M iz virtualne preslikava se opet na 0 u glavnoj
- ... (itd) ...

Opcenito: stranica s indeksom "i" u virtualnoj memoriji preslikava se u glavnoj memoriji na stranicu s indeksom: j = i%M (gdje "%" oznacava "modulo", odnosno ostatak cjelobrojnog dijeljenja).

Nas u zadatku zanima upravo taj "j", odnosno indeks stranice u glavnoj memoriji. Dakle prvo trebamo odrediti "i" za naš podatak, tj. indeks u virtualnoj memoriji, a onda cemo lako doci do "j" na temelju gornjeg izraza.

Mi pristupamo podatku s adresom 4098, a on je na stranici s indeksom i=4; evo zasto:
Posto je velicina stranice jednaka 1KB = 1024B, to ce vrijediti:
- stranica 0: podatci 0000 - 1023
- stranica 1: podatci 1024 - 2047
- stranica 2: podatci 2048 - 3071
- stranica 3: podatci 3072 - 4095
- stranica 4: podatci 4096 - 5119, a tu je upravo i naš podatak 4098

Opcenito: podatak se nalazi u virtualnoj memoriji na stranici i=adresa/velicina_stranice, gdje "/" oznacava operaciju cjelobrojnog dijeljenja.

6.

(5 bodova) Nacrtajte jednostavan model adresnog preslikavanje kojeg je predložio P. J. Denning. Ukažite na nelogičnost ili namjerno ugrađenu pogrešku u modelu te opišite ukratko način na koji se ta nelogičnost rješava. Rješenje prikažite za sljedeće parametre:

LAP = 32 M riječi,
FAP = 512 K riječi.

Broj straničnih okvira neka je 512. Odredite format virtualne (logičke) adrese, broj stranica, veličinu stranice te veličinu tablice preslikavanja.

Ono sto sigurno znam je da s obzirom da se radi o Denningu, velicina tablice je jednaka LAP-u, odnosno to je 32 M rijeci.

Ovo "rijeci" znaci da mnozimo velicinu adresnog prostora sa 4 B, jer jedna rijec ima 32 bita, tj 4 B, pa onda to prvo pretvorimo, i ispadne da je :

LAP = 32 M * 4 B = 32 * 2^20 * 4 B = 2^(5+20+2) B = 2^27 B = 128 MB,
FAP = 512 K * 4 B = 512 * 2^10 * 4 B = 2^(9+10+2) B = 2^21 B = 2 MB.

FAP ima stranicne okvire, dok LAP ima stranice. Velicina stranicnog okvira je jednaka velicini stranice, ali broj stranicnih okvira nije jednak broju stranica, vec je manji od broja stranica!

velicina FAP-a / broj stranicnih okvira = velicina jednog stranicnog okvira

2 MB / 512 = 4 KB = velicina jednog stranicnog okvira = velicina jedne stranice

velicina LAP-a / broj stranica = velicina jedne stranice

velicinu LAP-a imamo, ona iznosi 128 MB

velicinu jedne stranice takodjer imamo, ona pak iznosi 4 KB, kao i velicina jednog stranicnog okvira

pa dobijemo da je velicina LAP-a / velicina jedne stranice = broj stranica

128 MB / 4 KB = 32 768 stranica (broj stranica je veci od broja stranicnih okvira, broj okvira je zadan u zadatku, i iznosi 512)

Rekla sam na pocetku da je velicina tablice jednaka velicini LAP-a, tako da je velicina tablice = 128 MB.

I jos ostaje odrediti format virtualne (logicke) adrese...

VA = VS (virtualna stranica) + VP (virtualan pomak)

stranica je velicine 4 KB = 4096 B -> b = 4096

w(p) = log2 (4096) = 12 -> VP = 12 bita

w(o) = log2 (128 MB / 4 KB) = 15 -> VS = 15 bita

VA je dakle formata :

od 0. do 11. bita je adresni pomak, od 12. do 26. je adresna oznaka, odnosno od 0. do 14. adresna oznaka, a od 15. do 26. adresni pomak.

10.

(3 boda) Kapacitet primarne memorije je 512 M bajta a kapacitet sekundarne memorije je 32 G bajta. Memorijski sustav računala temelji se na virtualnoj memoriji sa straničenjem u kojem je stranica veličine 4 K bajta. Odredite indeks ili indekse straničnih priključaka na koji se može priključiti stranica čija je virtualna adresa 6194305 (dekadno). Virtualni memorijski sustav koristi tehniku izravnog memorijskog preslikavanja.

primarna 512 MB
sekundarna 32 GB
stranica 4 KB

n= 512 MB / 4 KB = 2^17 = 131072 stranica

j=6194305/4096=1512 -> adresa **bloka u sekundarnoj memoriji**

i=j mod n =1512 mod 131072 = 1512 -> indeks **stranice u glavnoj memoriji**

11.

(3 boda) Za računalo koje koristi virtualni memorijski sustav imamo sljedeće podatke:

- sekundarna memorija kapaciteta 16 G bajtova;
- primarna (radna) memorija kapaciteta 256 M bajtova;
- stranica kapaciteta 4096 bajtova;
- adresna zrnatost memorije je bajt;
- sustav koristi potpuno asocijativno preslikavanje.

Uz pretpostavku da je sustav jednokorisnički, treba odrediti:

a) format virtualne adrese;
b) format fizičke adrese;
c) broj bločnih priključaka;
d) veličinu tablice koja podržava adresnu translaciju (uz pretpostavku jednostavne izvedbe tablice).

Analizom nad grupom ispitnih programa pokazalo se da je omjer promašaja $0,6\%$, a pri tomu je bilo ukupno $2,8 \cdot 10^6$ referenciranja. Odredite broj pogodaka za gornje podatke.

formati VA i FA su isti... imas viseznacajne bitove gdje ti se nalazi adresna oznaka i manjeznacajni bitovi gdje se nalazi pomak. pomak je isti i kod VA i FA

dakle krenut cemo od toga

w(p) = log2b = log2 4096 = 12
dakle imamo 12 bita za pomak

e a oznaku dobijemo ovako :

kod FA:
w(o) = log2 (256MB/4kB) = 16

a kod VA:
w(o) = log2(16GB/4kb) = 22

dakle formati FA i VA izgledaju ovak:

a)
VA:
od 0 do 11 bita je adresni pomak, od 12 do 33 je adresna oznaka.

b)
FA:
od 0 do 11 bita je adresni pomak, od 12 do 27 je adresna oznaka.

c)Bp je velicina stranice koju dobimo tako da podjelimo velicinu FA i broj linija posto je S= n * b

dakle Bp = 256MB/4kb = 64kB

d)
sekundarna memorija je 16Gbajta = 16 x 2na30 => 34 bita potrebna
stranica 4096 bajta => 12 bit potrebna
velicina tablice koja podrzava adresnu translaciju je 2na 34 / 2na 12 = 2na 22 = 4M rijeci...

a omjer podogtka se dobije tako da se podjeli broj pogotka kroz broj referenciranja. posto je 0.6% omjer promasaja, onda je 99.4% omjer pogotka.

a broj pogotka = omjer pogotka * broj referenciranja.
ili broj pogotka = 0.994 * 2,8 * 10^6 = 2783200

http://materijali.fer2.net/File.7714.aspx

1. a) t1=1.8667ms, t2= 1ms --> Cpu2 ima bolje performanse

oni postoci ti trebaju za izracun globalnog CPI-a...

CPI (glob) ti se racuna tak da sumiras umnoske CPI i postotka instrukcija za taj CPI u odnosu na ukupni broj instrukcija...

dakle za I1 ce globalni CPI bit 0.1*1+0.2*2+0.5*3+0.2*4=2.8...
-0.1 je udio broja instrukcija za proces A spram ukupnog broja instrukc...dakle ni(a)/ni = 0.1, ni(a) je dakle 10^5, al nam to nije bitno za zad... da su nam zadali da proces A ima 10^5 instrukcija onda bi trebalo izracunat ovih 0.1...
-analogno tome su ovi drugi postoci

I2 glob CPI ce ti onda bit = 2

proc vrijeme == tcpu se racuna kao nc/f
broj taktova == nc=ni*CPI(glob)
--> 10^6*2.8/1.5Ghz za I1 --> 1.8667ms
--> 10^6*2/2Ghz za I2 --> 1ms

b) imas CPI(glob) na pocetku...
i sad trazis kak se taj globalni promjenio ako smo umjesto 4 ciklusa po instrukciji stavili 2 za proc D...
pa ce bit CPI(novo)=CPI(glob)-0.2*4+0.2*2=2.4

isto tak za C ak smo 3 zamijenili s 2...
CPI(novo)=CPI(glob)-0.5*3+0.5*2=2.3

isplatljivije je preinaciti C...

ovo mozes napravit i tak da si 2 nove tablice napravis di ces u jednoj imat drugu vrijednost za D, a u drugoj za C, pa postupak ide kao pod a)...

b)CPI(novo D)=2.4, CPI (novo C) = 2,3 --> promjena CPI(C) bi bila opravdanija

2. ovaj sam nekaj muljavio jer mi nije skroz jasan...
al sam si zamislio da nv=20 znaci da vektorsi proc obradi 20 instrukcija dok ovaj skalarni obradi 1...pa bi vektorski bio 20x brzi
s=20
x=?
a) p=2 (100% ubrzanje = 2 puta brze), isad se Amdahlov zakon (p=1/((1-x)+(x/s)) koristi i ispadne --> x=0.5263
b) p=10 (pola od maximalnog zanci 10x brze) --> x=0.947
c) p=2, s=100 (jer je nv=100) --> x=0.505

3. op kod = de --> 11011110
ra = r23 --> 010111
rb = r23 --> 010111
c2 = -9 --> 111111110111

rjesenje: 11011110|010111|010111|111111110111

4. ako instrukcija treba bit dugacka 16 bita a trebamo sve operacije zadrzat onda ce ovak valjda bit:
op kod=5
ra=2
rb=2
rc=2
c2=5

registre sam uzeo po 2 bita, jer ako se uzme vise, ostaje samo 2 bita za kostantu, a to je malo neprakticno...
ovak mozemo koristit 4 registra (adreseiramo ih s 2 bita, pa su moguce 4 adrese)...
konstanta ce nam imat 5 bita...
ovo ce smanjit kolicnu registara, smanjit direktan pristup memorijskim lokacijama...
trebat ce vise naredbi za neke stvari napravit...
bit ce potrebno pazljivije baratanje registrima...

5. prema sadrzaju memorije vidimo da treba napravit la r1, C2(r3) --> u registar r1 spremit vrijednost C2 + R[r3] (zbroj konstante i vrijednosti koja se nalazi u r3)..

C2 = 100A0 --> treba prosirit predznak, pa dobijemo FFFF00A0
r3 = 05000005

C2+r3 = 04FF00A5 --> ovo spremamo u r1

6. opet isto, samo kaj je sad u rb nula, pa radimo la r3,C2
spremamo u r3 vrijednost C2

C2 = 10001 --> prosirenjem to je FFFF0001
i sad spremimo C2 u r3...

7. a) la r7, 32 --> spremamo dekadskih 32 u r7, tj hexa 00000020 stavimo u r7
b) la r7, 32 (r5), uz to da je sadrzaj r5 = 510 (valjda dekadski)
sad u r7 spremamo C2+ sadrzaj r5

C2=00000020
r5=000001FE

C2+r5=0000021E --> spremamo u r7

8. a) radimo ld --> u ra spremamo sadrzaj koji se u memoriji nalazi na lokaciji (C2+rb)
za nas zadatak imamo ld r1,C2(r4)

C2=00000021
r4 =00000004

C2+r4 = 25 --> u r1 spremamo ono sto je u memoriji na 00000025 (nemamo zadano zadatkom sto je tamo, pa napisemo odgovor ovako)

b) la r1, C2(r4)
C2+r4=00000025
spremamo u r1 hexadekadski podatak 00000025...

9.

IF --> pribavljanje instrukcije iz cache-a, PC+4-->PC
ID --> dekodiranje instrukcije, pribavljanje registara, prosljedivanje konstanti
EX --> ALU operacije
MEM --> zapisivanje/citanje iz memorije
WB --> zapisivanje u registar

N>>M --> pretpostavljam da je to ASR za M bitova (CISC kod)

10.

model je onaj kaj je u predavanju broj 7...
vrijeme obrade jedne instrukcije je u neprotocnoj strukturi jednako 50ns
vrijeme obrade u protocnoj strukturi jednako je (broj segmenata svih instrukcija * trajanje jednog segmenta / N) --> (10^7+4)*10/10^7 = 10.000004ns
- 10^7+4 sam uzeo zato sto ako imamo jednu instrukciju, ona ima 5 segmenata, 2 instrukc imaju 6 segmenata, itd...tj. broj segmenata = broj instrukc + 4

odnos neprotoc/protoc = 50/10.0000004=4.999998 ...dakle protocni model je skoro 5x brzi

11.

imamo 40ns i 45ns...

vrijeme obrade jedne instrukcije je u neprotocnoj strukturi jednako (3*40+4*45) = 300ns

svaki segment protocne ce biti 45ns --> 7. predavanje 21 str. (uzima se trajanje najveceg kod protocne)
vrijeme obrade u protocnoj strukturi jednako je (broj segmenata svih instrukcija * trajanje jednog segmenta / N) --> (10000+6)*45/10000 = 45.027ns
- 10000+6 sam uzeo zato sto ako imamo jednu instrukciju, ona ima 7 segmenata, 2 instrukc imaju 8 segmenata, itd...tj. broj segmenata = broj instrukc + 6

odnos neprotoc/protoc = 300/45.027=6.6627...

12.

INC M (CISC)

RISC:

ld r1, M
addi r2, r1, 1 (RAW hazard - citamo r1, nakon sto smo zapisivali u njega)
st r2, M (RAW hazard - citamo r2, nakon sto smo u njega zapisivali)

segmenata po instrukc ima 4...
dakle

1._ _ _ _ (ld)
2. .._ _ _ _ (nop zbog RAW)
3. .. .._ _ _ _ (addi)
4. .. .. .._ _ _ _ (nop zbog RAW)
5. .. .._ _ _ _ (st)

broj perioda = 8

13.

n=12000, nema hazarda

za protocnu ce nam trajanje segmenta bit 18ns...

a) (n+4)*ts/n --> (12004*18/12000) = 18.006ns
ovo +4 je uzeto jer ima 5 segmenata po instrukc kao u prethodnim zadacima

b) ti=76ns za neprotocnu
neprotocna: 76ns
protocna: 18.006ns
omjer=76/18.006=4.2208 puta brze

14.

skicirat na temelju predavanja 7...
po meni je to naradba JMP x...

IF --> dohvati instrukciju, PC+4
ID --> dekodiraj instrukciju, proslijedi x
EX --> PC+4+x
MEM --> nista
WB -->upisi novi PC

bonus ne znam...

15.

latencija protocnog=broj segmenata*trajanje najduljeg segmenta (isto za svaku naredbu)
latencija neprotocnog-->samo zbroj trajanja segmenata koje te naredba sadrzi (razlicito ovisno o naredbi)

dakle:

neprotocno:

ld (ima svih 5 segmenata) --> latencija=625ps
st (nema WB segment) --> latencija=525ps
sub (nema MEM) --> latencija=475ps
addi (nema MEM) --> latencija=475ps

protocno:

najdulje trajanje jest 150ps (IF i MEM)
svaka naredba ima latenciju 5*150=750ps

16.

r1<--M[40+r6]
r6<--r2+r2 (WAR - pisanje u r6, nakon sto je koristen r6 za citanje)
M[50+r1]<--r6 (RAW - citamo r6 nakon sto smo u njega nesto upisivali)
r5<--M[r5-16]
M[r5-16]<--r5 (RAW - citamo r5 nakon upisivanja, WAR - upisujemo na [r5-16] nakon sto smo ga citali)
r5<--r5+r5 (WAR - upisujemo u r5, nakon sto smo ga prethodno citali)

Izmedju 1. i 3. instrukcije postoji RAW r1
Izmedju 4. i 6. RAW, WAR i WAW r6.

17.

opet ne znam dok ne dobijem odgovor na mejl kak tocno ovo treba gledat...
al po meni se vjeroajtno gleda da svaka perioda traje isto...

a) perioda ce trajat ko max segment --> 80ns...
perioda ce biti N+3 (jer je 4 segmenta) --> 10003...

trajanje jedne instrukcije = 80*10003/10000=80.024ns...

b) N-->beskonacno...
e sad mislim da mozemo uzet da je trajanje instrukcije = 80ns, a ne 80.024..
-kod jako velikog broja instrukcija prosj trajanje jedne se priblizava 80, a kod beskonacno je upravo = 80ns...

neprotocno: 40+70+50+80 = 240
protocno: 80

ubrzanje: 240/80 = 3x...

18.

a) prosjecno vrijeme = (10*(N1+7)+Trekonfig+10*(N2+7))/(N1+N2) = 10.00667ns

b) S=64/10.00667 = 6.3957x

19.

ovo bubam, ne nam jel to moguce tak izvest..

1. naredba:
tam di Rs ulazi u alu spojimo zbrajalo...pa izlaz zbrajala spojimo na reg cache...

2.naredba:
Rd spojimo na izlaz r2 iz reg cache-a, pa tak on moze doc u ALU...

20.

a)
```
do{
a=b;
a+=25;
b=a;
b+=4;
}while(b != c);
```

b)
nema pojma...

c)
neam pojma...

4. zadatak na 2. MI 2003

a) Virtualna adresa:

VA -> virtualna adresa
VS -> indeks virtualne stranice
VP -> pomak unutar stranice

VA = VS + VP

pošto je veličina stranice 4kB, a u memoriji je bajtna zrnatost, to znači da stranica ima 4k lokacija po 1B, dakle broj bitova koji je potreban da se odredi pomak unutar jedne stranice je 12, jer je 2^12=4096.

Virtualna adresa je takva da može adresirati cijelu sekundarnu memoriju, dajke 16GB = 16 * 2^30 = 2 ^34, dakle 34 bitova virtualne adrese.

VS = VA - VP = 22b.

b) Fizička adresa:

Kod virtualnog preslikavanja VP = FP, a za indeks virtualne stranice se pristupa straničnoj tablici u kojoj treba provjeriti da li za generiranu virtualnu adresu postoji fizička adresa.

fizička adresa se odnosi na RAM, dakle moramo s fizičkom adresom adresirati cijeli ram, 256MB = 2^28.
Iz toga slijedi da je FS(indeks fizičke stranice) = FA - FS = 16b.

c) broj straničnih priključaka se računa koliko možemo stranica smjestiti u RAM, 2^28 / 2^12, dakle kapacitet rama/veličina stranice. Ovdje se to može promatrati na sličan način i kao cache. Mi ustvari u ramu cacheiramo sadržaj sekundarne memorije, stvarajući privid rama veličine sekundarne memorije.

d) Stranična tablica:
u straničnoj tablica mora imati toliko riječi, da može pridjeliti svakom elementu virtualne memorije neku adresu. pošto je virtualna adresa 34 bitna i raspodijeljena je na okvire koji se indeksiraju s 22 bita = 4M zapisa.
Pitanje je kolko je svaki zapis velik. Ako uzmemo sliku sa slajda, za AMD opteron, ima 6b koji su servisni + 16bitova koji određuju indeks okvira, tj stranice u ramu = 24 b = 3B, pa je ukupna veličina tablice 4M * 3B = 12MB, što je uostalom i rješenje tog zadatka.
http://www.fer.unizg.hr/_download/re...naMemorija.pdf slajd 16, po tome sam računao veličinu pojedinog zapisa u straničnoj tablici.

5. zadatak ZI 2011/2012

Mislim da je najbolje krenuti od oblika adrese.
12 bitova je indeks imenika (imenik je prva razina) -> to nam kaže koliko imenik ima elemenata, 2 na 12.
10 bitova je indeks tablice (svaki element imenika pokazuje na jednu tablicu ili ni na jednu, ako nam ne treba; tablica je druga razina) -> to nam kaže koliko jedna tablica ima elemenata, 2 na 10
10 bitova je offset u stranici -> to nam kaže veličinu jedne stranice

Sad, krenimo ovako.
Zrnatost je bajtna (podrazumjeva se, rekao bih). Mogućih offseta ima 2^10. 2^10 puta bajtna zrnatost = 1KB - veličina jedne stranice. 1024 hex je 400.

Program i statički podatci se rasprostire preko ABC adresa. Koliko najmanje puta moramo uzeti 400 da bude veće of ABC? 3 puta. Znači, 3 stranice trebamo za program i statičke podatke.
Stog se rasprostire preko 521 adresa (hex). Za to nam trebaju 2 stranice.
Dinamičko zauzimanje nema ulogu, odnosno samo nam je bitno koliko je zauzeto i na kojoj adresi.
Zauzeto je 1KB, za što je dovoljna jedna stranica.

Sve skupa koristimo 3+2+1 stranica. Iz adresa gdje se stranice nalaze se vidi da na prve 3 stranice pokazuje jedna tablica (adrese su jedna do druge i stanu u jednu tablicu), na druge dvije koje su opet jedna do druge pokazuje još jedna tablica i na zadnju isto pokazuje jedna tablica.

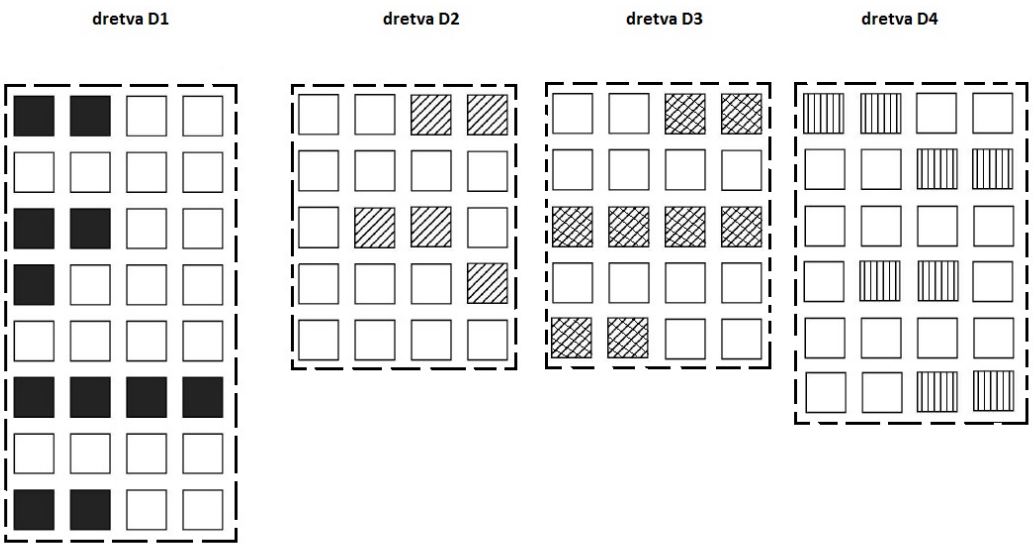
Svaka tablica mora imati svoj broj elemenata, a svaki je unos velik 4B. Dakle,
4B×3tablice×1024=12KB
Imenik ima 4096 elemenata, svaki po 4B. To je 16KB.
Tablice i imenik zauzimaju ukupno 28KB.

Ostaju nam adrese. Imenik počinje na 0x1000, te zauzima 4096(elemenata)×4B(veličina jednog elementa) slijednih lokacija.
Prvih 12 bita svake adrese je indeks imenika. Ostaje ti za svaki indeks jednostavnom matematikom izračunati gdje se nalazi. Indeks×4B(veličina jednog elementa)+0x1000.

1 DRETVE

Zadane su četiri dretve D1, D2, D3 i D4 (slika) koje se izvršavaju na procesoru sa četiri protočne strukture (pretpostavlja se da su protočne strukture visoko specijalizirane) prikazati izvođenje dretvi D1-D4 za:

a) superskalarni procesor
b) visedretverni (MT)
c) simultani visedreveni (SMT)



RJEŠENJE:

Zadatak 4. sa ovogodisnjeg ZI je iz zadnjeg predavanja (str. 49. - str 59.).
Zadana slika na ispitu prikazuje instrukcije pojedinih dretvi (svako polje odgovara jednoj instrukciji), pri cemu stupci odgovaraju protocnim strukturama (ima ih 4), a retci vremenskim periodama. Napomena da su protocne strukture visoko specijalizirane govori nam da se instrukcije moraju izvoditi u stupcu u kojem se nalaze (ne moze se instrukcija iz prvog stupca izvoditi u drugom ili trecem stupcu).

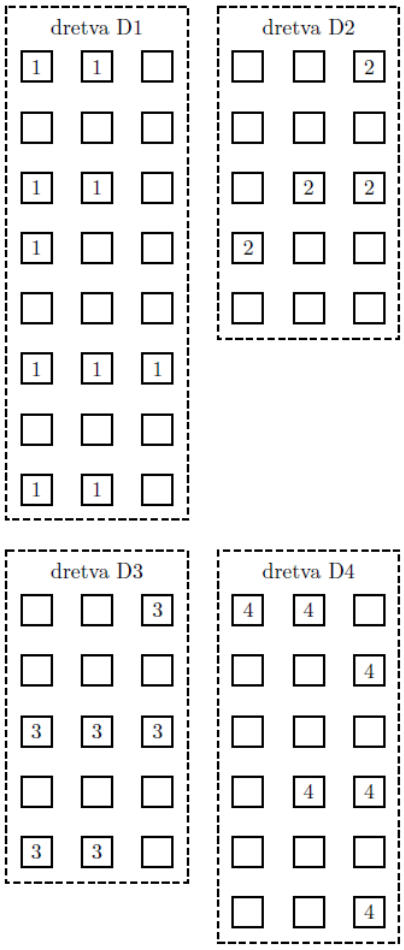
Superskalarni procesor moze izvoditi vise instrukcija istovremeno (kako su vec zadane slikom), ali te instruckije moraju pripadati jednoj dretvi. Takoder, superskalarni procesor izvesti ce cijelu dretvu D1, pa zatim dretvu D2, pa D3 i onda D4. Dakle, nece mijesati instrukcije pojedinih dretvi. Vremenske periode koje su prazne (nema instrukcija), ostati ce prazne.

Visedretveni procesor ce rijesiti problem praznih vremenskih perioda. On ce u prvoj vremenskoj periodi izvesti instrukcije iz prvog retka D1. U drugoj iz prvog retka D2, u trecoj iz prvog retka D3, u cetvrtoj iz prvog retka D4, u petoj iz drugog redak (u ovom slucaju treceg jer je drugi redak prazan) i tako sve dok ne izvede instrukcije svih dretvi. Dakle, procesor naizmjence izvodi instrukcije zadanih dretvi, ali unutar jedne periode izvodi instruckije samo jedne dretve.
1		1				
				2		2
				3		3
4		4				
1		1				
		2		2		
....
Procesor koji podrzava simultanu visedretvenost za razliku od MT procesora, moze unutar jedne vremenske periode izvoditi instrukcije vise razlicitih dretvi. Tako za zadani primjer unutar prve periode procesor ce izvesti 2 instrukcije od prve dretve unutar prva dva stupca (protocne strukture). Zatim primjetiti ce da su mu ostala jos dva prazna stupca unutar kojih bi mogao smjestiti instrukcije drugih dretvi, pa ce pogledati dretvu D2. Dretva D2 ima dvije instukcije u 3 i 4 stupcu, te njih moze smjestiti u prvi redak sa instrukcijama dretve D1. Time je prvi redak biti popunjen. Zatim procesor uzima instrucije iz dretve D3 (3. i 4. stupac) te ih smjesta u drugi redak, pa gleda dretvu D4 koja ima 2 instrukcije u 1. i 2. retku. Njih takoder smjesta u drugi redak. Time je i drugi redak popunjen. Procesor krece sa trecim retkom i ponovo gleda dretvu D1 (zadnja je bila D4) i uzima njene instrucije.
|1| |1| |2| |2|
|4| |4| |3| |3|
|1| |1| |4| |4|

|1| |2| |2| | |
|3| |3| |3| |3|
...
Bitno je zapamtiti da procesor unutar jedne periode moze izvoditi instrukcije vise razlicitih dretvi, te da skupine instrukcija jedne dretve koje se izvode u jednoj periodi ne moze "lomiti" (npr. ako pise |3| |3| |3| | | onda se zadane 3 instrucije moraju izvesti zajedno, ne moze |3| | | |3| | |, pa zatim | | |3| | | | |).

6. (10 bodova) Zadane su _cetiri dretve D1, D2, D3 i D4 (slika (b)) koje se izvr_savaju na procesoru s tri proto_cne strukture (pretpostavlja se da su proto_cne strukture visoko specijalizirane). Prikazati izvodenje dretvi D1-D4:
(a) Za superskalarni model procesora
(b) Za vi_sedretveni (_nozrnati) model procesora (MT)

(c) Za model procesora koji podr_zava simultanu vi_sedretvenost (SMT)



(b) (Slika uz zadatak 6.) Instrukcije dretvi D1-D4

RJEŠENJE:
Dakle 6. zadatak sa ZI prosle godine:
MT finozrnati model -> promjena konteksta dogada se u svakoj periodi pri cemu se dretve odabiru slijedno u krug (round robin) (sve su istog prioriteta)
11*
**2
**3
44*
11*

*22
333
**4
...
Na sto treba paziti? Kada bi dretva D1 izgledala ovako:
D1
11*
***
***
***
***
11*
onda u petoj periodi ne bi mogli pisati 11* jer se time razmak
izmedu prve i seste periode dretve D1 smanjio za 1 sto nije u redu.
U tom slucaju jednostavno bi preskocili dretvu D1 i presli na dretvu
D2.
SMT - u jednoj periodi mogu se izvesti instrukcije razlicitih dretvi
Protocne strukture su specijalizirane sto znaci ako za dretvu
D1 pise
11* ona ce i u SMT modelu te instrukcije morati pisati bas tako
(nece moci biti 1*1 ili *11).
Ako strukture nisu specijalizirane, onda ovo prethodno ne vrijedi,
tj. instrukcije se mogu unutar periode smjesiti u bilo koji
slobodni prikljucak. Npr. ako imamo
D1
11*
i D2
22*
u SMT modelu ih mozemo napisati kao
112
2** (presli smo u novu periodu zato sto nam vise nije stalo u
prethodnoj)
Rjesenje iz ZI bi bilo:
112
443
114
*22
333
144
2**
111
334
11*

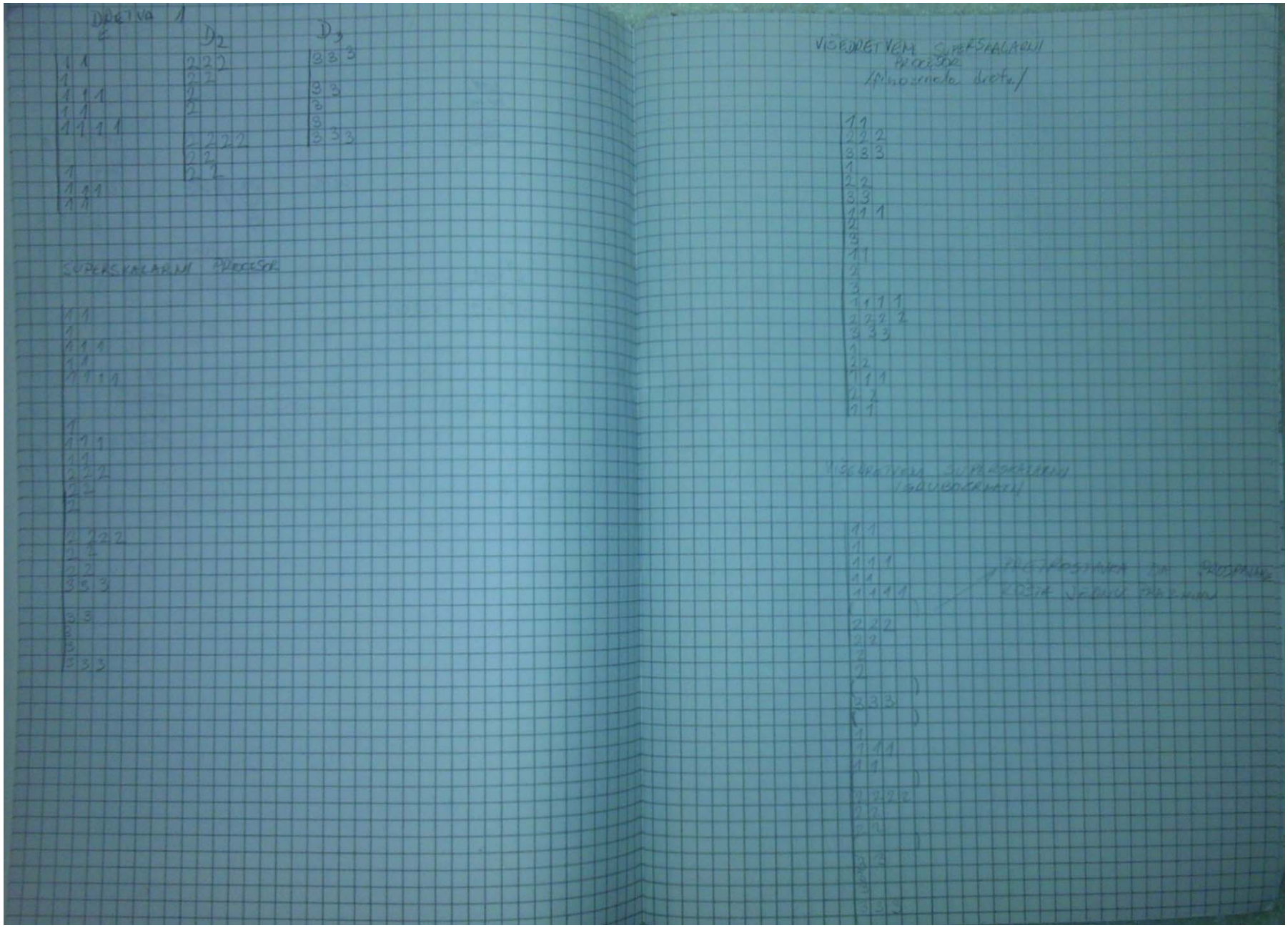
U grubozrnatom modelu MT promjena konteksta se dogada kada se dogodi
duzi zastoj (npr. zbog promasaja L2). U ovom zadatku to nije pitanje pa bi
rjesenje izgledalo slicno superskalarnom procesoru:
cijela dretva D1
jedna perioda za promjenu konteksta
cijela D2
jos jedna prazna perioda
D3
prazno
D4
Ako bi na pola dretve D1 bio promasaj L1 onda bi se tada dogodila promjena konteksta:
Prvih pola D1

prazno
cijela D2
prazno
D3
prazno
D4
prazno
drugih pola D1

Ovaj zadatak opet nije dovoljno slikovit za slucaj kada protocne strukture nisu specijalizirane no kada bi dretve izgledale ovako:
D1
11*
1**
D2
22*
22*
D3
3**
***
33*
onda bi SMT izgledao ovako:
112
231
22*
33*

Lp,
BS

Primjer sa sata od kolege


DREVO 1
D_2
D_3
SUPERSKALARNI PROCESOR
VIŠEDREVESNI SUPERSKALARNI PROCESOR
(simultana drevesa)
VEČJEDREVESNI SUPERSKALARNI PROCESOR
(SMT)

Malo objašnjenja: Dobiju se u zadatku 3 dretve recimo i onda se mora vidjet za svaki superskalarni procesor kako se ponaša s izmjenom dretvi.

Važno: Svaki procesor ima četiri "stupca" tj. mjesta u jednom retku za staviti dretvu. Kako sam prepisivao nisam imao vremena crtati rešetke...ali skuzit cete.

1. Superskalarni procesor: dretve idu po redu uključujuči rupe. Kad jedna dretva Završi odmah se nastavlja druga (nema rupa zbog promjene konteksta)

2. Višedretveni superskalarni - finozrnati. Stavljate dretve naizmjence, 1-2-3-1-2-3 itd. Što kad dođe rupa u nekoj dretvi - tu rupu popunjavaju ostale dretve. Kad opet dođe na red dretva u kojoj se pojavila rupa normalno se nastavlja izmjenjivanje dretvi. (bitno je samo da dretva koja čeka ciklus tj. ima rupu, idući ciklus i odčeka)

3. Vd.ss. - grubozrnati. Stalno vrti prvu dretvu dok ne dođe do rupe. Onda napravi rupu za promjenu konteksta i stavlja drugu dretvu. Znači ne mijenja ih kao u finozrnatom svaki ciklus nego tek kad dođe rupa.

4. Simultano višedretveni procesor ili SMT. Što je fora kod njega? Da popuni ne samo rupe nego i sva mjesta u jednom retku, tj. sve stupce koje procesor moze u jednom ciklusu obradit. On je zeznutiji malo pa cu ga posebno opisati (pod uvjetom da sam ga dobro shvatio jer je to Ribarić nešto smuljao na kraju sata). Ispravite me ako ima potrebe.

Dretve su na gornjoj slici.

1122
2333
122[] - rupa na 3. dretvi, a ne možemo ponovno na jedinicu jer je u ovom ciklusu iskorištena
1112
3311
2311
113 - bila rupa na dvojci (a s iduceg ciklusa nije mogla doći jer mora čekati jedan ciklus nakon prošle dvojke (gore))
2222
3332
21 - trojke nema više, a dvojku ne smijemo jer još koristimo onu tekućeg ciklusa (ne smijemo miješati dretvu)
2211
1 - isto objašnjenje kao i prethodno
11 - kraj

Što je cilj zadatka? **Pokazati kako u procesoru optimizirati vrijeme potrebno da se tri dretve izvrte.**

Dopisivanje kolegice u vezi dretvi
Nisam bio na satu kada je profesor objasnjavao zadatak, no Vase rjesenje odgovara slucaju kada *protocne strukture* (ne dretve)

nisu specijalizirane. U ispitu koji spominjete pise da jesu.

Ako kojim slucajem imate knjigu Patterson&Hennessy, objasnjenje koje je tamo ponudeno vrijedi upravo za zadatak sa sata.

Ako ne, mozda vam ovaj pdf malo pomogne:

https://web.cs.dal.ca/~mheywood/CSCI...ipe/10-TLP.pdf

ili pak ovaj (cijela knjiga, proslo izdanje, str.174.)
http://citeseerx.ist.psu.edu/viewdoc...=rep1&type=pdf

Lp,
BS

-----Original Message-----

Dakle zelite reci da je sam zadatak bio postavljen na drugaciji nacin, odnosno u ispitu su dretve bile specijalizirane, a na satu ne?

PS. moje isprike, bila sam mijenjala dijelove teksta, pa sam obrisala malo previse a ostavila krivo :/

-----Original Message-----

To nije isti primjer.

Ono sto Vas mozda muci je kako iz ovog:
D1 D2 D3

11 222 333

dobijemo ovo:

SMT:
1122
2333

To je zato sto u ovom slucaju protocne strukture (stupci) nisu specijalizirane, tako da mozete instrukciju zadane dretve postaviti u bilo koji prazan stupac (promatranog retka).
Stavimo 11 -> 1122(ne stane nam vise; nova perioda) ->
1122 -> 1122
2 2333

Kada imate specijalizirane strukture, instrukcija mora stajati bas u onom stupcu u kojem je stajala i u promatranoj dretvi.

Ostala objasnjenja sa rupama vrijede: ako je razmak izmedu 2 instrukcije n perioda u promatranoj dretvi, razmak izmedu tih instrukcija kada ih stavimo u SMT ne smije biti manji od n perioda. Ako bi ipak bio, ostavljamo prazno i nastane rupa.

Lp,
BS

PS: prof. Ribaric

-----Original Message-----

Postovanje!

Imam pitanje vezano uz zadatak sa dretvama.
Naime, prosle je godine na zavrsnom ispitu bio takav zadatak pa su Vama slali pitanje kako se rijesava (kod pripremanja za ispitni rok), a Vas je odgovor bio slijedeci:

"Procesor koji podrzava simultanu visedretvenost za razliku od MT procesora, moze unutar jedne vremenske periode izvoditi instrukcije vise razlicitih dretvi. Tako za zadani primjer unutar prve periode procesor ce izvesti 2 instrukcije od prve dretve unutar prva dva stupca (protocne strukture). Zatim primjetiti ce da su mu ostala jos dva prazna stupca unutar kojih bi mogao smjestiti instrukcije drugih dretvi, pa ce pogledati dretvu D2. Dretva D2 ima dvije instukcije u 3 i 4 stupcu, te njih moze smjestiti u prvi redak sa instrukcijama dretve D1. Time je prvi redak biti popunjen. Zatim procesor uzima instrucije iz dretve D3 (3. i 4. stupac) te ih smjesta u drugi redak, pa gleda dretvu D4 koja ima 2 instrukcije u 1. i 2. retku. Njih takoder smjesta u drugi redak. Time je i drugi redak popunjen. Procesor krece sa trecim retkom i ponovo gleda dretvu D1 (zadnja je bila D4) i uzima njene instrucije.

|1| |1| |2| |2|
|4| |4| |3| |3|
|1| |1| |4| |4|
|1| |2| |2| | |
|3| |3| |3| |3|"

Mene zanima zasto je onda kolega Ribaric na satu ovaj primjer rijesio ovako (takoder za smt):
D1 D2 D3

11 222 333
1 22 ////
111 2 33
11 2 3
//// //// 3
//// 2222 3333
1 22
1111 22
11

SMT:
1122
2333
122[] - rupa na 3. dretvi, a ne mozemo ponovno na jedinicu jer je u ovom ciklusu iskoristena
1112
3311
2311
113 - bila rupa na dvojci (a s iduceg ciklusa nije mogla doci jer mora cekati jedan ciklus nakon prosle dvojke (gore))
2222
3332
21 - trojke nema vise, a dvojku ne smijemo jer jos koristimo onu tekuceg ciklusa (ne smijemo mijesati dretvu)
2211
1 - isto objasnjenje kao i prethodno
11 - kraj

Unaprijed hvala!

Lijep pozdrav