

*Електротехнички факултет Универзитет у Београду* 15.05.2021.

*Катедра за рачунарску технику и информатику*

#### Архитектура и организација рачунара 2

#### – колоквијум –

**1. (5)** Објаснити технику оптимизације рада кеш меморије која користи више банки унутар кеш меморије (*Multibanked Caches*). Шта је потребно да постоји код процесора и код кеш меморије да би дата техника могла да се примени.

**2. (5)** У процесору се налази *L1 Data* асоцијативна кеш меморија величине 512KiB, која користи *write-through* алгоритам за ажурирање садржаја оперативне меморије са *no write allocated* политиком довлачења. Користи се LRU алгоритам замене. Величина блока кеш меморије је 64B.

Дат је програм који исписује за сваки дан месеца маја колика је измерена највиша температура у том месецу.

У матрици **temp** се налазе реалне вредности типа double (ширине 64 бита) које представљају измерену температуру за мај месец за сваки дан за сваки секунд у дану. Матрица **temp** има 31 колону (колона 0 – представља податке за 1. мај, колона 1 – представља податке за 2. мај итд.) и 86400 редова података (за сваки секунд у дану). Претпоставити да матрица **temp** на почетку није учитана у кеш меморију, као и да је прва ћелија матрице **temp[0][0]** поравната са блоком кеш меморије. Компајлер матрицу смешта по редовима.

**for** (**register int** i = 0; i < 31; i = i + 1){

**register double** max=MIN\_FLOAT;

**for** (**register int** j = 0; j < 86400; j = j + 1)

**if** (max < temp[j][i])

max = temp[j][i];

printf("maj %d.:", max);

}

Сматрати да приступ податку који се налази у кеш меморији траје 2 ns, а довлачење целог блока кеш меморије из оперативне меморије траје 50 ns. Уколико је дошло до кеш промашаја, прво се из оперативне меморије учита цео блок у кеш меморију, па се тек онда приступа траженом податку у кеш меморији.

а) Потребно је оптимизовати дати код тако да се резултат програма не промени, а да се при томе искористе карактеристике дате кеш меморије. Није дозвољено коришћење наменских инструкција за манипулацијом кеш меморије. У програму је доступно за коришћење још 10 регистара опште намене.

б) Израчунати укупно време које је потребно за приступ подацима матрице **temp** (анализирати део кода без исписа). Сматрати да услов max < temp[j][i] ће бити испуњен сваки 10 пут. Дати време за дати код, као и време за оптимизовани код (из ставке по а) )

**3. (5)** Код хардверске виртуализације користи се техника пратеће табела страница (*Shadow Page Tables*). Описати ову технику. Дати пример пресликавања једне виртуелне адресе госта у физичку адресу домаћина и потпунити све потребне табеле.

**4. (5)** Потребно је делове датог програма оптимизовати. Програм учитава низ студената и смешта их у меморију. Сматрати да је почетна адреса низа за чување студената поравната са блоком кеша. Над низом примењује метода **Student\*** pronadjiStudentaPoIndeksu(**Student\*** list, **int** n, **char[]** gggg, **char[]** bbbb). Метода враћа показивач на активног студента који има индекс gggg/bbbb, параметар list је показивач на листу студената; n представља број студената у листи; gggg и bbbb су низови дужине 4 карактера. Величина података су:

sizeof(int) = sizeof(float) = 32 bit; sizeof(bool) = sizeof(char)=8 bit, адресе се су ширине 32 бита.   
Програм се извршава на процесору који поседује кеш меморију чија је величина блока 128 бита.Остале карактерситике кеш меморије нису доступне.

**struct** Student {

**int** id;

**int** jmbg;

**char** gggg[4];

**char** bbbb[4];

**char** pol;

**char\*** Ime;

**char\*** Prezime;

**int** godinaStudija;

**float** prosek;

**bool** isActive;

**char** polozenoIspita;

**char** nivoStudija;

}

...

**Student\*** pronadjiStudentaPoIndeksu(**Student\*** list, **int** n, **const** **char[]** gggg, **const char[]** bbbb) {

**for** (**int** i=0; i<n; i++){

**bool** indeks\_ok=true;

**for**(**int** j=0;j<4 && indeks\_ok;j++)

indeks\_ok = indeks\_ok && list[i].gggg[j]==gggg[j];

**for**(**int** j=0;j<4 && indeks\_ok;j++)

indeks\_ok = indeks\_ok && list[i].bbbb[j]==bbbb[j];

**if** (indeks\_ok && list[i].isActive)

**return** list[i];

}

**return** 0;

}

**int** main(){

...

**int** n;

**Student**\* studenti;

procitajStudente(studenti, &n);

...

**Student\*** s = pronadjiStudentaPoIndeksu(stidenti, n, "2000", "0001");

...

}

a) Предложити нову структуру Student, тако да се дата метода pronadjiStudentaPoIndeksu ефикасније извршава.

б) Није могуће изменити структуру Student. Потребно је оптимизовати дату методу pronadjiStudentaPoIndeksu трансформацијом и убацивањем инструкције prefetch на одговарајућим местима тако да не постоји ни један промашај у кеш меморији. Претпоставити да је резултат инструкције prefetch видљив након извршених 5 редова датог кода (prefetch писати у засебно у новом реду).

в) Трансформисати методу pronadjiStudentaPoIndeksu коришћењем Loop fusion начина оптимизације кода.

**Напомене:** На колоквијуму нису дозвољена никаква помоћна средства, ни калкулатори ни литература. Колоквијум траје **90 минута**.