

ЛПРС2 Домаћи

Манипулисање Битима

верзија 1.0

Милош Суботић, Стефан Пијетловић

21. април 2020.

1 Домаћи

Пре започињања домаћег задатка, прочитати ову датотеку до краја.

Домаћи задатак је извршити одређене манипулације битима над меморијски мапира-ним регистрима замишљене улазно-излазна јединице. Детаљи спецификације за сваког студента су дефинисани у спецификацији (датотека **Specification.pdf**). Сваки студент у приложеној табели ће наћи свој број студента, и помоћу тог броја ће наћи своју спецификацију у горепоменутој датотеци. У овом документу су описани општи детаљи, који важе за сваког студента.

Јединица има 4 32-битна меморијска регистра, под називом X, Y, Z и W. Регистар X има 4 подрегистра ширина дефинисаних у спецификацији. Регистар Y има низ од 2 подрегистра, сваки по 16 бита. Регистар Z има низ од 4 подрегистра, сваки по 8 бита. Регистар W нема подрегистара. Мапа је дата испод:

1. X (32b)

- A (2-4b)
- B (4-8b)
- C (6-12b)
- D (8-20b)

2. Y (32b)

- M[4] (4x8b)

3. Z (32b)

- N[2] (2x16b)

4. W (32b)

Редослед регистара X, Y, Z и W не мора да буде као у овој листи, већ је исто дефинисан спецификацијом. Ширине A, B, C и D регистара су исто дефинисане у спецификацији. Операције које требају да се изврше над регистрима су такође дефинисане у спецификацији. Користити макрое кад год је то могуће. У неким операцијама ће бити потребно урадити конкатенацију преко померања и битског или.

Објашњење о инсталацији и покретању примера дано је у **LPRS2_Lab_Bit_Manipulation** пројекту, у датотеци **Bit_Manipulation.pdf**, верзија 1.1.

Резултат домаћег задатка треба да буде:

1. Цртеж регистарске мапе На основу редоследа и ширина регистара нацртати регистарску мапу, по примеру из лекције `Bit_Manipulation`, с том разликом да треба игнорисати апсолутне адресе. Цртеж регистарске мапе може бити:

- Слика може бити нацртана руком па сликана,
- нацртана па експортирана у слику путем неког програма за цртање (Paint, Gimp...),
- табела у xls или doc.
- или просто искористите `LATEX` из лекције `Bit_Manipulation`.

Цртеж ставите у Docs фасциклу.

2. `main.c` код са урађеним TODO-овима, по узору на лекцију `Bit_Manipulation`:

- Макрои,
- битска поља (започето):
 - са регистрима,
 - са унијама, а свака да има:
 - * регистар `r`,
 - * структуру са подрегистрима `s`,
 - * структуру са распакованим подрегистрима `b`,
- алиаси (`p8`, `p32`, `pr`, `pu`),
- операције са исписима.

3. `Makefile` са попуњеним личним подацима.

- Број студента дефинисан у приложеној табели,
- број индекса,
- име и презиме.

Дозвољено је имати **код и исписе** који **нису захтевани спецификацијом** а служе за **дебаговање и експериментисање**, али их је потребно искључити након завршетка задатка. Неуредна варијанта је коментарисати га. Боља варијанта је користити предпрофесор ради укључивања и искључивања дебаг мода. Дебагерски и експериментали **код** је потребно **ставити између `#if DEBUG` и `#endif`**, а за дебагерске **исписе** користити `dbg_printf` као на излисту испод.

```
1 #if DEBUG
2     // Experiment: Is my mask good?
3     printf("A2 = %d\n", pu->x.b.a2);
4     p32[X] ^= 1<<A2;
5     printf("A2 = %d\n", pu->x.b.a2);
6     p32[X] ^= 1<<A2;
7     printf("A2 = %d\n", pu->x.b.a2);
8 #endif
9     dbg_printf("A2 = %d\n", pu->x.b.a2); // Before set.
10    p32[X] |= 1<<A2;
11    dbg_printf("A2 = %d\n", pu->x.b.a2); // After set.
12
13    printf("A = 0x%01x\n", pu->x.s.a);
```

Да би укључили дебаг мод потребно је при покретању **make** команде проследити 1 **DEBUG** променљивој, као на излисту испод.

1

```
make DEBUG=1
```

Ако се проследи 0 **DEBUG** променљивој, или се пак додела изостави, дебаг мод ће бити искључен.

Све TODO-ове који су одрађени обришите. Сав код да је поравнат. Грешке (error) и упозорења (warning) при компајлирању се такође бодују, негативно.

На самом крају, кад је све готово само у фасцикли пројекта само извршите команду:

1

```
make dist
```

која ће обрисати бинарне датотеке програма и запаковаће се zip датотека са решеним задатком. Тај zip онда подигните да Сову.