Задания за практическа работа по
Компютърни архитектури
за специалност Информатика, редовно, II курс
през 2023-2024 учебна година

Указания общо по цялото задание

Изпълнението на това задание дава 70 точки от възможни общо 140 (останалите 70 точки се получават от различни изпитвания върху лекционния материал). На базата на получените точки (от 0 до максимално възможните 140) ще се формира окончателната оценка по учебния предмет Компютърни архитектури.

Решението на това задание трябва да бъде предадено чрез качване в класната стая за упражненията по Компютърни архитектури (т. е. през същия сайт, от който е получено това задание).

В деня на изпита по всяка задача ***ще бъдат задавани въпроси*** *за проверка дали студентът разбира представените решения, а също така, дали може да изпълнява малки модификации на решенията в съответствие с такива промени в условията на задачите, които запазват основните идеи и на исканото от задачата, и на нейното решение.*

От всеки студент се изисква да реши изцяло предназначените за неговия факултетен номер три задачи. Разпределението на задачите според факултетните номера е описано в първата таблица по-надолу.

Указания по първа задача

Тази задача включва четири подзадачи със самостоятелни и независими едно от друго решения.

За представяне на решението на първа задача са допустими всякакви електронни формати, в това число и снимки, получени от съответен хартиен носител (може те да се окажат най-удобни за отчитане на това задание).

В решенията непременно трябва да присъствуват (в уместен кратък вид) междинните изчисления, водещи до резултата. При оценяването ще бъдат приемани само решения, в които крайният резултат наистина се получава от представените междинни изчисления.

Указания по втора задача

Като решение на втора задача трябва да бъде качен през сайта на класната стая файл, който съдържа решението на съответната задача, във вид на асемблерен код (очаква се файлът с разширение `.asm`*).*

Представяните приложения трябва да бъдат написани именно на асемблерния език NASM, а въвеждането и извеждането на данните трябва да става точно чрез макросите, предоставени от интегрираната среда SASM.

Указания по трета задача

По тази задача също трябва да бъде качен файл. Файлът трябва да съдържа програмния код на език от високо ниво на приложението за съответния вариант на трета задача. Езикът от високо ниво се избира от студента и може да бъде само един от C, C++, Java, C# или Free Pascal.

Разпределение на задачите

Последната цифра на факултетния номер на студента определя еднозначно кои задачи трябва да реши този студент, както описва следната таблица:

Номера на задачи, които трябва да реши студентът	*За студентите с кои факултетни номера са предназначени задачите с такива номера*
или 1.1 и 1.2, или 1.1 и 1.3	*факултетни номера, завършващи с цифрата* ***1***
или 2.1 и 2.2, или 2.1 и 2.3	*факултетни номера, завършващи с цифрата* ***2***
или 3.1 и 3.2, или 3.1 и 3.3	*факултетни номера, завършващи с цифрата* ***3***
или 4.1 и 4.2, или 4.1 и 4.3	*факултетни номера, завършващи с цифрата* ***4***
или 5.1 и 5.2, или 5.1 и 5.3	*факултетни номера, завършващи с цифрата* ***5***
или 6.1 и 6.2, или 6.1 и 6.3	*факултетни номера, завършващи с цифрата* ***6***
или 7.1 и 7.2, или 7.1 и 7.3	*факултетни номера, завършващи с цифрата* ***7***
или 8.1 и 8.2, или 8.1 и 8.3	*факултетни номера, завършващи с цифрата* ***8***
или 9.1 и 9.2, или 9.1 и 9.3	*факултетни номера, завършващи с цифрата* ***9***
или 10.1 и 10.2, или 10.1 и 10.3	*факултетни номера, завършващи с цифрата* ***0***

Задачи

Първа задача
(това са задачите 1.1, 2.1, 3.1, 4.1, 5.1, 6.1, 7.1, 8.1, 9.1 и 10.1)

Общо относно задачите 1.1, 2.1, 3.1, 4.1, 5.1, 6.1, 7.1, 8.1, 9.1 и 10.1

Номерът на конкретната задача, която студентът трябва да реши, е назован в таблицата по-горе.

Всяка от тези задачи включва четирите подзадачи (означени по-надолу като Подзадача А, Подзадача Б, Подзадача В и Подзадача Г), които се решават независимо една от друга.

Всяка подзадача еднозначно се определя от едно от четири числа N_1, N_2, N_3 и N_4. Тези означения са използвани по-надолу в общото условие на първа задача. Всяка конкретна подзадача се получава чрез заместване на съответното число, едно от N_1, N_2, N_3 и N_4, което участвува в условието на тази подзадача.

Съответствието между номер на задача и стойностите на четирите числа N_1, N_2, N_3 и N_4, използвани в общото условие за първа задача, описва следната таблица:

Номер на задача	N_1 в **десетичен** запис	N_2 в **двоичен** запис	N_3 в **десетичен** запис	N_4 в **двоичен** запис
1.1	-816	100010111	-22,325	101101010000
2.1	-562	100101001	-59,3125	101001100000
3.1	-530	101011011	-21,925	101010110000
4.1	-768	100101011	15,975	101001001000
5.1	-898	100001001	71,1125	101011001000
6.1	-930	110001001	-40,725	100111100000
7.1	-616	100111011	59,125	101101001000
8.1	-586	101011001	-49,375	001001110000
9.1	-656	100100101	58,1125	001001010000
10.1	-774	101011011	17,125	100111110000

Общо условие на първата задача
(т. е. общо условие за задачите 1.1, 2.1, 3.1, 4.1, 5.1, 6.1, 7.1, 8.1, 9.1 и 10.1)

Подзадача А

Да се напишат **двоичните записи** на **11-разрядните прав, обратен и допълнителен кодове** на числото N_1.

Освен отговорите, да се опишат кратко междинните изчисления, даващи крайния резултат.

(Конкретната подзадача, която трябва да реши един конкретен студент, се получава при заместването на N_1 със стойността, взета от втората колона на горната таблица от реда, започващ с номера на задача, която трябва да реши студентът.)

Подзадача Б

Да се напишат **десетичните записи** на числата $\boldsymbol{x}$, $\boldsymbol{y}$ и $\boldsymbol{z}$, за които е изпълнено условието:

$$\text{ПК}_9(\boldsymbol{x}) = \text{ОК}_9(\boldsymbol{y}) = \text{ДК}_9(\boldsymbol{z}) = N_2$$

Освен отговорите, да се опишат кратко междинните изчисления, даващи крайния резултат.

(Конкретната подзадача, която трябва да реши един конкретен студент, се получава при заместването на N_2 със стойността, взета от третата колона на горната таблица от реда, започващ с номера на задача, която трябва да реши студентът.)

Подзадача В

Да се напише **двоичният запис** на $\text{НКПЗ}_{5,8}\left(N_3\right)$.

В искания код трябва всички цифри от мантисата, включително първата, да се съхраняват в паметта, а позицията на дробната запетайка се подразбира точно след първата съхранявана цифра.

Освен отговорите, да се опишат кратко междинните изчисления, даващи крайния резултат.

(Конкретната подзадача, която трябва да реши един конкретен студент, се получава при заместването на N_3 със стойността, взета от четвъртата колона на горната таблица от реда, започващ с номера на задача, която трябва да реши студентът.)

Подзадача Г

Да се напише **десетичният запис** на онова число $\boldsymbol{K}$, което е кодирано точно, без закръгляне чрез следния код с плаваща запетая: $\text{КПЗ}_{4,7}(\boldsymbol{K}) = N_4$.

В дадения код всички цифри от мантисата, включително първата, действително се съхраняват в паметта, а позицията на дробната запетайка се подразбира точно след първата съхранявана цифра.

Освен отговорите, да се опишат кратко междинните изчисления, даващи крайния резултат.

(Конкретната задача, която трябва да реши един конкретен студент, се получава при заместването на N_4 със стойността, взета от петата, последната колона на горната таблица от реда, започващ с номера на задача, която трябва да реши студентът.)

Втора задача
(това са задачите 2.1, 2.2, 3.2, 4.2, 5.2, 6.2, 7.2, 8.2, 9.2 и 10.2)

Номерът на конкретната задача, която студентът трябва да реши, зависи от последната цифра на факултетния номер на студента, както това съответствие е описано в първата таблица (на страница 2) по-горе в настоящия текст.

Пояснение общо за задачите 1.2, 2.2, 3.2, 4.2, 5.2, 6.2, 7.2, 8.2, 9.2 и 10.2

Решението на такава задача, както и всяко писане на асемблерен език, съществено се улеснява при уместно използване на:

— добро структуриране на кода;

— мнемонични имена за етикетите;

— подходящи коментари (удобни, информативни, без излишъци, добре структурирани и т. н.);

— отделяне като самостоятелни процедури на онези части от кода, които или се използват многократно (когато има такива), или по някакъв критерий са с повишена сложност (за писане или за обмисляне);

— удобно съгласуване на използваните регистри и данни;

— избор на такива средства за построяване на алгоритъма, които по-просто, по-директно или по-ефективно водят до желаната цел.

При оценяването на представените от студентите решения по втора задача, спазването на тези норми на добър стил на програмиране ще бъде основание за повишаване на оценката. Съответно нарушаването на добрия стил на програмиране няма да бъде основание за намаляване на оценката (разбира се, то излишно, но закономерно, ще затруднява работата на студента).

Задача 2.1

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа NK със стойност цяло положително число, по-малко от 11.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете NK пъти числа, които едновременно са четни и не се делят на осем.

в) Извежда или максималното прочетено число, което е по-малко от (NK-10).2, или съобщение, че няма нито едно такова число измежду прочетените.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които едновременно са четни и не се делят на осем.

Пример

стойност на константата NK:

```
2
```

входен поток:

```
30  31  -1  -9  7  26
```

изходен поток:

```
maximum: no numbers
store: 30 26
```

Пример

стойност на константата NK:

```
6
```

входен поток:

```
4  -4  23  -8  20  -16  24  24
```

изходен поток:

```
maximum: -16
store: 4 -4 20
```

Задача 2.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа RT със стойност цяло положително число, по-малко от 30.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете RT пъти числа, които едновременно се делят на 4 и са по-големи от RT-40.

в) Извежда едно от прочетените числа, които имат минимална абсолютна стойност.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които едновременно се делят на 4 и са по-големи от RT-40.

Пример

стойност на константата RT:

```
25
```

входен поток:

```
-21  -22  -26  -14  -20  -20
```

изходен поток:

```
with min abs: -14
store:
```

Пример

стойност на константата RT:

```
6
```

входен поток:

```
22  16  7  25  -42  -12  23  20  19  26  26
```

изходен поток:

```
with min abs: 7
store: 16 -12 20
```

Задача 3.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа ML със стойност цяло положително число, по-малко от 10.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете ML пъти числа, които или са нечетни, или имат абсолютна стойността, по-малка от 10+ML.

в) Извежда броя на числата, които едновременно се делят на четири и са по-малки от 3.ML-30.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които или са нечетни, или имат абсолютна стойността, по-малка от 10+ML.

Пример

стойност на константата ML:

```
4
```

входен поток:

```
14  -34  34  -4  29  -32  -3  26  34  0
```

изходен поток:

```
number: 1
store: -4 29 -3 0
```

Пример

стойност на константата ML:

```
2
```

входен поток:

```
-12  29  4
```

изходен поток:

```
number: 0
store: 29 4
```

Задача 4.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа HS със стойност цяло положително число, по-малко от 15.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете HS пъти числа, които едновременно са по-големи от HS-100 и са по-малки или равни на HS+20.

в) Извежда броя на прочетените числа, които са по-големи от 5 и дават остатък, по-голям от три, при делене с осем.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които едновременно са по-големи от HS-100 и са по-малки или равни на HS+20.

Пример

стойност на константата HS:

```
4
```

входен поток:

```
-33  -25  -86  81  -11
```

изходен поток:

```
number: 0
store: -33 -25 -86 -11
```

Пример

стойност на константата HS:

```
5
```

входен поток:

```
86  104  -61  -87  -97  94  -43  -66  97  -140  -69
```

изходен поток:

```
number: 2
store: -61 -87 -43 -66 -69
```

Задача 5.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа K55 със стойност цяло положително число, по-малко от 10.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете K55 пъти числа, които или са по-малки от K55-20, или са по-големи от K55+10.

в) Извежда максималния остатък, който се получава при делене с 16 на абсолютната стойност на някое прочетено число.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които или са по-малки от K55-20, или са по-големи от K55+10.

Пример

стойност на константата K55:

```
2
```

входен поток:

```
-28 20
```

изходен поток:

```
maximum: 12
store: -28 20
```

Пример

стойност на константата K55:

```
3
```

входен поток:

```
6  0  -16  1  -27  -6  8  11  13  13
```

изходен поток:

```
maximum: 13
store: -27
```

Задача 6.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа N48 със стойност цяло положително число, по-малко от 16.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете N48 пъти числа, които едновременно са по-големи от N48+5 и дават остатък 5 при делене с осем.

в) Извежда или минималното прочетено число, по-голямо от N48-7, или съобщение, че няма прочетено нито едно такова число

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които едновременно са по-големи от N48+5 и дават остатък 5 при делене с осем.

Пример

стойност на константата N48:

```
2
```

входен поток:

```
37  3 -1  45
```

изходен поток:

```
minimum: -1
store: 37 45
```

Пример

стойност на константата N48:

```
1
```

входен поток:

```
47  -8  -10  59  31  -3  20  -8  24  -1  7  -1  0  9  21
```

изходен поток:

```
minimum: -3
store: 21
```

Задача 7.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа LL със стойност цяло положително число, по-малко от 20.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете LL пъти числа, които едновременно са по-големи от LL+10 и дават остатък три при делене с четири.

в) Извежда броя на нечетните прочетени числа, които са по-малки от LL-5.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които едновременно са по-големи от LL+10 и дават остатък три при делене с четири.

Пример

стойност на константата LL:

```
1
```

входен поток:

```
16  11  -11  -9  6  30  -2  -17  15
```

изходен поток:

```
amount: 3
store: 15
```

Пример

стойност на константата LL:

```
6
```

входен поток:

```
-8  -8
```

изходен поток:

```
amount: 0
store:
```

Задача 8.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа W8 със стойност цяло положително число, по-малко от 12.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете W8 пъти числа, които едновременно са по-големи от 20-W8 и дават остатък, по-голям от 2, при делене с осем.

в) Извежда броя на прочетените числа, които се делят на четири и едновременно с това са по-малки от W8.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които едновременно са по-големи от 20-W8 и дават остатък, по-голям от 2, при делене с осем.

Пример

стойност на константата W8:

```
2
```

входен поток:

```
11  10  3  15  19  0  17  22
```

изходен поток:

```
amount: 1
store: 19 22
```

Пример

стойност на константата W8:

```
5
```

входен поток:

```
-1  19  -1  21  12  -10  7  -6  17  12  3  3
```

изходен поток:

```
amount: 0
store: 19 21
```

Задача 9.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа RLJ със стойност цяло положително число, по-малко от 20.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете RLJ пъти числа, които едновременно са по-големи от RLJ-10 и дават остатък, различен от три и от пет, при делене с осем.

в) Извежда броя на прочитанията (колко пъти е прочетено) на максималното прочетено число, което е по-малко от -RLJ.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които едновременно са по-големи от RLJ-10 и дават остатък, различен от три и от пет, при делене с осем.

Пример

стойност на константата RLJ:

```
6
```

входен поток:

```
-4  -6  -1  -11  26  -2  -34  -18  -24  -28  5  7  7
```

изходен поток:

```
maximum: -11
store: -1 26 -2 7 7
```

Пример

стойност на константата RLJ:

```
16
```

входен поток:

```
-2  23  13  -1  25  25
```

изходен поток:

```
maximum: no numbers
store: 23 25 25
```

Задача 10.2

Да се напише на асемблерния език NASM с използване на макросите за вход/изход, предоставяни от интегрираната среда SASM, приложение, което:

а) Съдържа декларация на именувана константа F4 със стойност цяло положително число, по-малко от 10.

б) Въвежда цели числа, докато или прочете два пъти подред едно и също число, или прочете F4 пъти числа, които, умножени по три, дават стойност, по-малка от удвоената стойност на F4.

в) Извежда броя на прочитанията (колко пъти е прочетено) на минималното прочетено число, което е по-голямо от -F4.

г) Извежда всички прочетени числа, по толкова пъти, по колкото са били прочетени, които, умножени по три, дават стойност, по-малка от удвоената стойност на F4.

Пример

стойност на константата F4:

```
4
```

входен поток:

```
7  9  6  9  9
```

изходен поток:

```
number of maximums: 3
store:
```

Пример

стойност на константата F4:

```
3
```

входен поток:

```
7  -6  6  -4  -13
```

изходен поток:

```
number of maximums: 1
store: -6 -4 -13
```

Трета задача
(това са задачите 1.3, 2.3, 3.3, 4.3, 5.3, 6.3, 7.3, 8.3, 9.3 и 10.3)

Номерът на конкретната задача, която студентът трябва да реши, зависи от последната цифра на факултетния номер на студента, както това съответствие е описано в първата таблица (на страница 2) по-горе в настоящия текст.

Общо условие за задачите 1.3, 2.3, 3.3, 4.3, 5.3, 6.3, 7.3, 8.3, 9.3 и 10.3

Дадено е приложение, написано на асемблерния език NASM с използване на макросите за въвеждане и извеждане на интегрираната среда SASM.

Да се напише на език от високо ниво такъв код, че при еднакви коректни входни данни асемблерното приложение и приложението на език от високо ниво да извеждат един и същи резултат. При въвеждане на некоректни данни е допустимо всякакво действие на приложението на език от високо ниво.

Езикът от високо ниво може да бъде всеки от `C, C++, Java, C#` или `Free Pascal`.

В следващата таблица за всеки номер на задача е посочено приложението на асемблерен език (т. е. файлът с асемблерен код), което трябва да бъде транслирано до език от високо ниво:

Название на файла с асемблерен код	***Номер на задача***
test_03_01.asm	***1.3***
test_03_02.asm	***2.3***
test_03_03.asm	***3.3***
test_03_04.asm	***4.3***
test_03_05.asm	***5.3***
test_03_06.asm	***6.3***
test_03_07.asm	***7.3***
test_03_08.asm	***8.3***
test_03_09.asm	***9.3***
test_03_10.asm	***10.3***

Асемблерният код на приложенията от тази таблица е извън настоящия текст. Файловете от лявата колонка на таблицата са приложени отделно като самостоятелни файлове в архива, съдържащ файла с настоящия текст.