Технически Университет - Габрово	**Организация на компютъра**
Тема: Микропроцесори – част 2	**Лабораторно № 2**
Цел: Анализ на изпълнението на програмен код. Представяне на данните в различни бройни системи	

I. Теоретична част

За да напишете програмен код за даден микропроцесор може да използвате различни начини за кодиране на желаната логика. Най-често се използват езици от високо ниво като Java, C# и C++. В този случай вие използвате синтаксисът на съответния език, за да реализирате необходимата програмна логика. Този синтаксис обаче е неразбираем за микропроцесорите. Той трябва да бъде преобразуван до последователност от инструкции за съответната архитектура, например Intel x86. Това се реализира от компилаторите (Java виртуалната машина при програмния език Java). Следователно, наборът от инструкции е специфичен за всеки процесор. Виртуалният CPU с който ще работим (*Wombat*) има 12 инструкции, които са описани в Табл. 1.

Табл. 1. Инструкции на виртуалния CPU

Инструкция	Предназначение	Пример
add	Добавя към съдържанието на акумулатора съдържанието на клетка от паметта и записва резултата в акумулатора	$add\ x$ acc+x -> acc
$subtract$	Изважда от съдържанието на акумулатора съдържанието на клетка от паметта и записва резултата в акумулатора	$subtract\ y$ acc-y -> acc
$multiply$	Умножава съдържанието на акумулатора със съдържанието на клетка от паметта и записва резултата в акумулатора	$multiplay\ x$ acc*x -> acc
$devide$	Дели съдържанието на акумулатора и съдържанието на клетка от паметта и записва резултата в акумулатора	$devide\ y$ acc/y -> acc
$read$	Чете цяло число от конзолата и записва в акумулатора	$read$ console -> acc
$write$	Показва съдържанието на акумулатора на конзолата	$write$ acc -> console
$load$	Чете съдържанието на клетка от паметта и го записва в акумулатора	$load\ sum$ [sum] -> acc
$store$	Записва съдържанието на акумулатора в клетка от паметта	$store\ sum$ acc -> [sum]
$jump$	Безусловен преход	$jump\ Label1$ Label1 address -> pc

$jmpz$	Преход при нулев резултат в акумулатора	$jmpz\ Label2$
		if (acc !=0) skip
		else
		Label2 address -> pc

$jmpn$	Преход при отрицателен резултат в акумулатора	$jmpz\ Label3$
		if (acc >=0) skip
		else
		Label3 address -> pc

$stop$	Установява бит HALT (спира изпълнението на инструкциите)	$stop$ bit HALT = 1

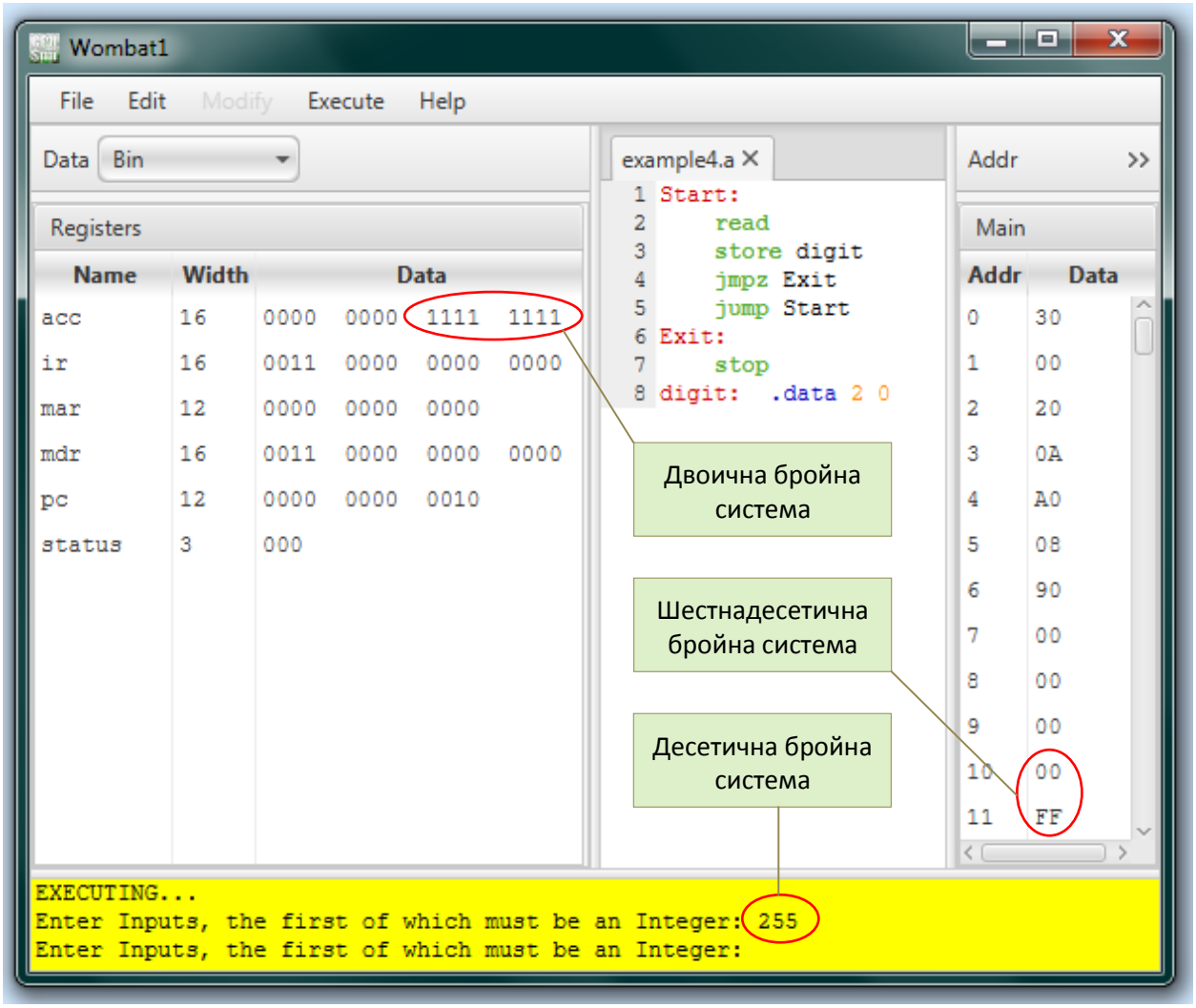
II. Задачи за изпълнение

Задача 1:

Напишете инструкциите които въвеждат число от конзолата. Числото да се записва в клетка от паметта с име $digit$. Изход от програмата – при въвеждане на 0.

Решение:

На Фиг. 1 е показано примерно решение на Задача 1 и текущото съдържание на регистрите и паметта.



Фиг. 1. Примерно решение на Задача 1

Задайте съдържанието на регистрите да се вижда в двоична бройна система, а съдържанието на паметта – в шестнадесетична бройна система. Така ще можете да

проверите всяко число, което въвеждате от конзолата в десетична бройна система как се представя в другите две бройни системи. Например 255 десетично е FF шестнадесетично и 1111 1111 двоично. Дефинирането на клетка от паметта се реализира чрез псевдо-инструкция $.data$. Синтаксисът ѝ изисква задаване на два параметъра: първият указва колко байта от паметта заема променливата (2), а вторият – каква е началната стойност на променливата (0).

Отговорете на следните въпроси:

- Кое е максималното положително и минималното отрицателно число, което може да въведете от конзолата?
- В какъв формат се записват числата в паметта: big-endian или little-endian?

Задача 2:
Напишете инструкциите които намират сумата на две числа, които се въвеждат от конзолата. Резултатът да се изведе на конзолата.

Решение:

За въвеждане на числата ще използваме инструкция $read$ (виж Фиг. 2). Ще запишем двете числа в клетки от паметта с имена x и y (ред 2 и ред 4). Ще съберем двете числа чрез инструкция add (ред 5). Резултатът се визуализира чрез инструкция $write$ (ред 6).

```
1 Start:  read          ; read a digit 1 -> acc
2         store x       ; acc -> x
3         read          ; read a diogit 2 -> acc
4         store y       ; acc -> y
5         add x         ; acc + x -> acc
6         write
7         stop
8 x:      .data 2 0     ; x
9 y:      .data 2 0     ; y
```

Фиг. 2. Примерно решение на Задача 2

Въпрос:

Кои редове от този програмен код можем да изтрием, без това да се отрази на крайния резултат?

Това решение на задачата използва запаметяване на двете числа в паметта, а на практика е необходимо запаметяване само на едното от тях (например x). Стойността на другото (y) няма смисъл да се запомня в паметта. Поради тази причина, ред 4 и ред 9 са излишни.

Задача 3:
Напишете инструкциите които се проверява дали едно число, въведено от конзолата, е равно, по-голямо или по-малко от предварително зададено число. Програмният код да показва на конзолата:

- 0, ако числата са равни;
- 1, ако въведеното число е по-голямо от зададеното;
- -1, ако въведеното число е по-малко от зададеното.

Помощ:

Печата на конзолата се реализира чрез инструкция $write$. Тя показва на конзолата текущото съдържание на акумулатора. Тъй като можем да зареждаме акумулатора само със съдържанието на клетка от паметта, то трябва да заделим памет за две променливи, чрез които ще изпълним заданието: едната ще е със стойност 1, а другата – със стойност -1. За да проверим дали въведеното число е равно, по-голямо или по-малко от зададеното (клетка от паметта) можем да използваме инструкция $subtract$.

Задача 4:
Проверете как се зареждат в паметта низове. За целта използвайте псевдо-инструкция $.ascii$, например:

$$.ascii\ „Hello\ world!\backslash n"$$

Колко байта се отделят за всеки символ? Как се кодират управляващите ASCII кодове \n, \r и \t?

III. Допълнителни въпроси

1. Каква е причината програмният броят на виртуалния CPU да е 12 битов, а не 16 битов.

2. Как може да реализирате бързо умножение и деление по степените на 2?

3. Възможно ли е чрез псевдо-инструкция $.ascii$ да дефинираме низ на кирилица? Обосновете отговора си.