School of Electrical and Electronic Engineering

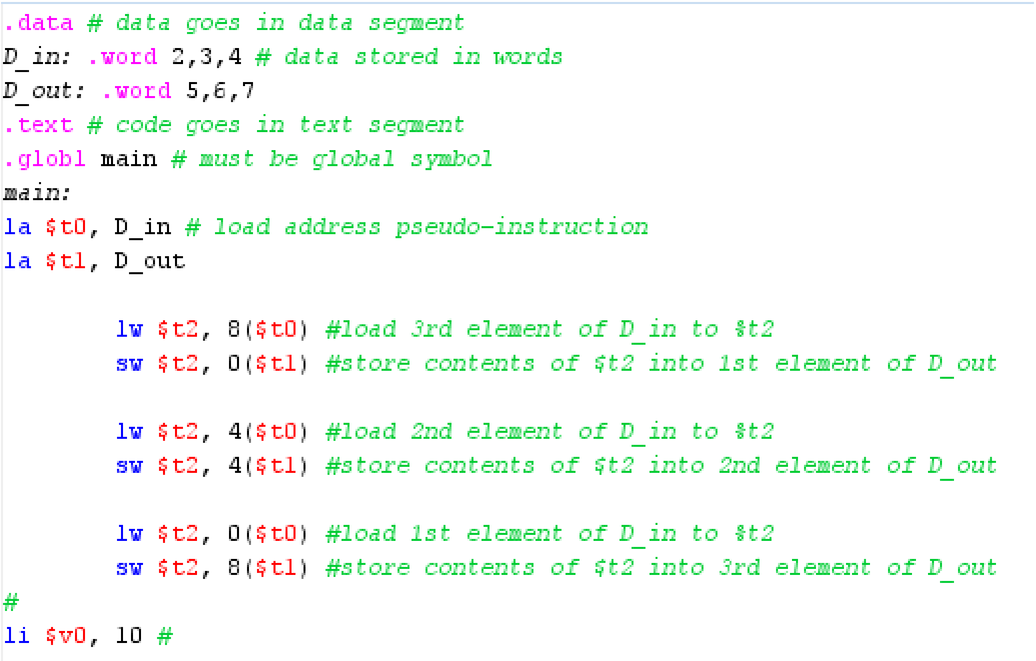
COMP 30080

Processor design

Student Name:	Paul Doherty
Student Number:	10387129
Date:	10/02/2014
Assignment:	Assignment 1

Q1 (a)

The function of the code presented in question 1 is a simple element overwriting code. Its function is to overwrite the contents of *D_out* by replacing its contents with *D_in* in the reverse order. So that the 1st element of the *D_out* array will be replaced with the 3rd element of the *D_in*, and the 2nd element of the *D_out* array will be replaced with the 2nd element of the *D_in* and finally the 3rd element of the *D_out* array will be replaced with the 1st element of the *D_in*. Figure 1 below shows the commented code below to achieve this.

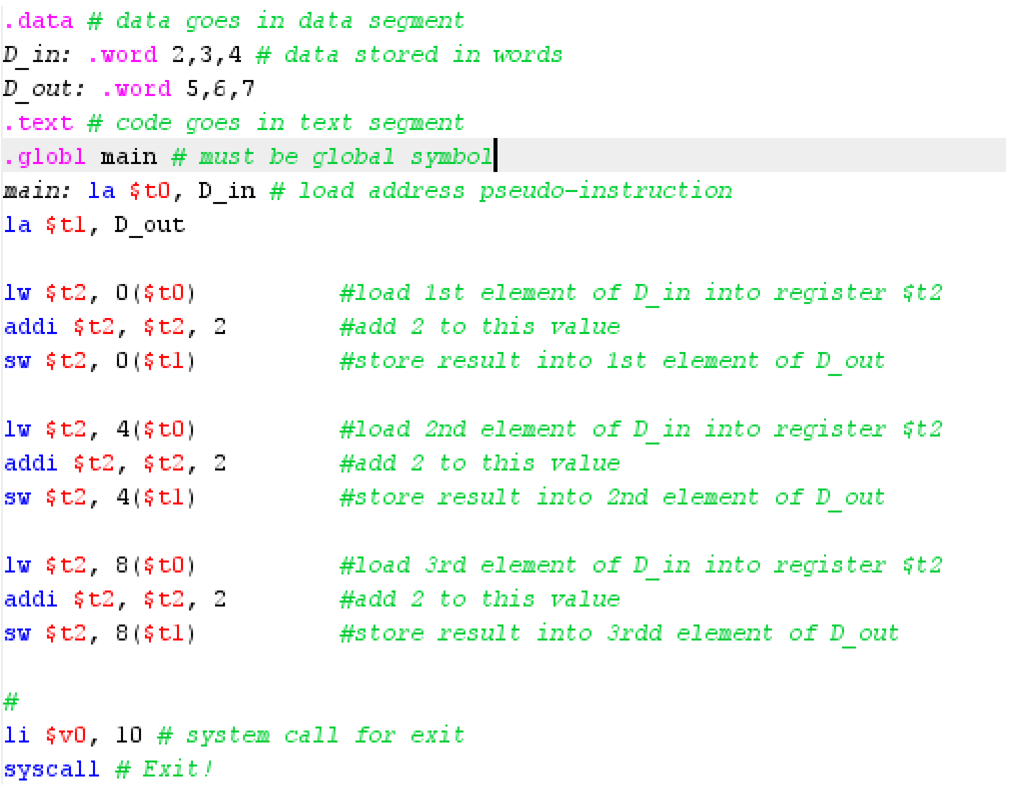


```
.data # data goes in data segment
D_in: .word 2,3,4 # data stored in words
D_out: .word 5,6,7
.text # code goes in text segment
.globl main # must be global symbol
main:
la $t0, D_in # load address pseudo-instruction
la $t1, D_out

        lw $t2, 8($t0) #load 3rd element of D_in to $t2
        sw $t2, 0($t1) #store contents of $t2 into 1st element of D_out

        lw $t2, 4($t0) #load 2nd element of D_in to $t2
        sw $t2, 4($t1) #store contents of $t2 into 2nd element of D_out

        lw $t2, 0($t0) #load 1st element of D_in to $t2
        sw $t2, 8($t1) #store contents of $t2 into 3rd element of D_out
#
li $v0, 10 #
```

Figure 1: Shows code for Question 1 (a) with added comments

Q1 (b)

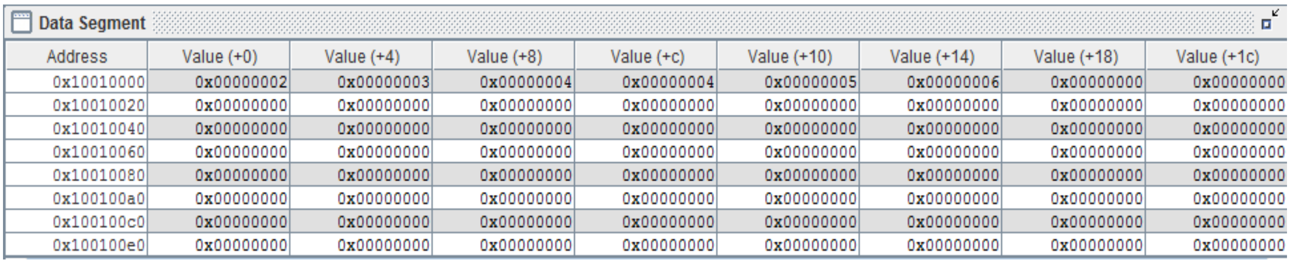
Figure 2 shows how the code was implemented:



```
.data # data goes in data segment
D_in: .word 2,3,4 # data stored in words
D_out: .word 5,6,7
.text # code goes in text segment
.globl main # must be global symbol
main: la $t0, D_in # load address pseudo-instruction
la $t1, D_out

lw $t2, 0($t0)          #load 1st element of D_in into register $t2
addi $t2, $t2, 2        #add 2 to this value
sw $t2, 0($t1)          #store result into 1st element of D_out

lw $t2, 4($t0)          #load 2nd element of D_in into register $t2
addi $t2, $t2, 2        #add 2 to this value
sw $t2, 4($t1)          #store result into 2nd element of D_out

lw $t2, 8($t0)          #load 3rd element of D_in into register $t2
addi $t2, $t2, 2        #add 2 to this value
sw $t2, 8($t1)          #store result into 3rdd element of D_out

#
li $v0, 10 # system call for exit
syscall # Exit!
```

Figure 2: Shows code used to implement Q1(b)

The output of this executed code can be seen in figure 3. It successfully shows *D_out* containing the values *D_in+2.*

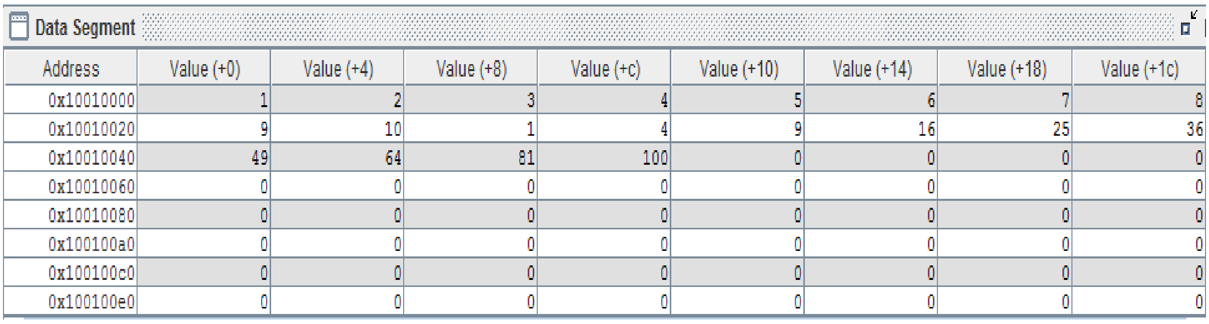


Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000002	0x00000003	0x00000004	0x00000004	0x00000005	0x00000006	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Figure 3: Shows data segment from executed code from figure 2.

Q2

This problem requires me to write a program that will calculate the squares of the numbers 1-10 using addition only by using a loop. The successful output of this code can be seen in figure 4.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	2	3	4	5	6	7	8
0x10010020	9	10	1	4	9	16	25	36
0x10010040	49	64	81	100	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0

Figure 4: This shows the output of the program that squares the numbers from 1-10

The program I wrote to achieved this has been uploaded to moodle and as *10387129_ass1_q2.asm.*

Q3

Question 3 required me to write a program that would determine if two lists matched of not; For example, the inputs 1, 2, 3, 4, 5 and 1, 2, 3, 4, 5 and 5 should give the output True. While, the inputs 0, 1, 2, 3, 4 and 1, 2, 3, 4, 5 and 5 should give the output False.

To view my full working program the file can be seen in moodle as *10387129_ass1_q3.asm.*