SAKARYA UNIVERSITY

Spring 2015

BSM 206 Computer Organization

Homework #2 Solutions

Questions:

1) [50points] [Computer Abstractions and Technology] In the next two pages, there are MIPS codes for two procedures: Search and Next, where the second one is called in the first one.

Search procedure uses argument registers $a0, $a1, $a2, and $a3 for integer m, base address of integer Array A, base address of integer Array B, and integer len, respectively. It also uses saved registers $s0 and $s1 for local integers r and i, respectively.It also uses $v0 for return value. Both Array A and B have length more than len.

Next procedure, on the other hand, uses argument registers $a0, $a1, and $a2 for integer i, base address of integer Array B, and integer len, respectively. It also uses saved registers $s0 and $s1 for local integers k and j, respectively.It also uses $v0 for return value.

Based on this information, please answer the following questions.

a) Please comment next to each instruction by explaining what it does. For Search on page 2, and for Next on page 3.
b) For the MIPS codes given for each instruction, provide the machine code as shown in the table following the MIPS codes. Please add as much rows as needed for the program. The memory address of the first instruction is given. Please write memory addresses for each instruction. The first instruction is given in R-format, please reshape the rows for different formats by merging cells. For Search on page 4, and for Next on page 5.
c) [Bonus: 20 points]Please write down the relevant high-level language code (either in C or Java) on the upper half of page 6, which provides these MIPS codes.

```
Label  Addr.  MIPS Code
Search:20000 addi   $sp,   $sp,   -12     # adjust stack for 3 items
       20004 sw     $ra,   8 ($sp)        # save return address
       20008 sw     $s1,   4 ($sp)        # save the value in s1 prior to procedure
       20012 sw     $s0,   0 ($sp)        # save the value in s1 prior to procedure
       20016 add    $s0,   $zero, $zero   # initialize r, (r = 0)
       20020 add    $s1,   $zero, $zero   # initialize i, (i = 0)
L1:    20024 slt    $t0,   $s1,   $a3     # if i < len, then t0 = 1, else t0 = 0
       20028 beq    $t0,   $zero, E2      # if t0 = 0 (i ≥ len), go to E2
       20032 sll    $t0,   $s1,   2       # t0 = i*4 (i < len)
       20036 add    $t0,   $t0,   $a1     # t0 = A + (i*4) (address of A[i]
       20040 lw     $t1,   0 ($t0)        # t1 = A[i]
       20044 bne    $t1,   $a0,   E1      # if (t1 ≠ m), go to E1
       20048 addi   $sp,   $sp,   -8      # adjust stack for 2 items
       20052 sw     $a1,   4 ($sp)        # save base address of A to stack
       20056 sw     $a0,   0 ($sp)        # save m to stack
       20060 add    $a0,   $s1,   $zero   # a0 = i
       20064 add    $a1,   $a2,   $zero   # a1 = base address of B
       20068 add    $a2,   $a3,   $zero   # a2 = len
       20072 jal    Next                  # call procedure Next
       20076 lw     $a1,   4 ($sp)        # a1 = base address of A
       20080 lw     $a0,   0 ($sp)        # a0 = m
       20084 addi   $sp,   $sp,   8       # pop 2 items from stack
       20088 add    $s0,   $v0,   $zero   # r = v0
       20092 j      E2                    # go to E2
E1:    20096 addi   $s1,   $s1,   1       # increment i
       20100 j      L1                    # go to L1
E2:    20104 add    $v0,   $s0,   $zero   # return r
       20108 lw     $ra,   8 ($sp)        # restore return address
       20112 lw     $s1,   4 ($sp)        # restore s1 prior to procedure
       20116 lw     $s0,   0 ($sp)        # restore s0 prior to procedure
       20120 addi   $sp,   $sp,   12      # pop 3 items from stack
       20124 jr     $ra                   # go to return address
```

```
Label  Addr.  MIPS Code
Next:  40000  addi  $sp,  $sp,   -8     # adjust stack for 2 items
       40004  sw    $s1,  4 ($sp)       # save the value in s1 (i in Search)
       40008  sw    $s0,  0 ($sp)       # save the value in s0 (r in Search)
       40012  add   $s0,  $zero, $zero  # initialize k, (k = 0)
       40016  add   $s1,  $zero, $zero  # initialize j, (j = 0)
L2:    40020  slt   $t0,  $s1,   $a2    # if j < len, t0 = 1, else t0 = 0
       40024  beq   $t0,  $zero, E4     # if t0 = 0 (j ≥ len), go to E4
       40028  sll   $t0,  $s1,   2      # t0 = j*4
       40032  add   $t0,  $t0,   $a1    # t0 = B + j*4 (address of B[j])
       40036  lw    $t1,  0 ($t0)       # t1 = B[j]
       40040  bne   $t1,  $a0,   E3     # if t1 ≠ i, go to E3
       40044  add   $s0,  $s1,   $zero  # k = j
       40048  j     E4                  # go to E4
E3:    40052  addi  $s1,  $s1,   1      # increment j
       40056  j     L2                  # go to L2
E4:    40060  add   $v0,  $s0,   $zero  # return k
       40064  lw    $s1,  4 ($sp)       # restore s1 (i)
       40068  lw    $s0,  0 ($sp)       # restore s0 (r)
       40072  addi  $sp,  $sp,   8      # pop 2 items from stack
       40076  jr    $ra                 # go to return address
```

Machine Code for Search

Memory	op	rs	rt	rd	shamt	funct
				constant or address		
20000	8	29	29	-12		
20004	43	29	31	8		
20008	43	29	17	4		
20012	43	29	16	0		
20016	0	0	0	16	0	32
20020	0	0	0	17	0	32
20024	0	17	7	8	0	42
20028	4	8	0	19		
20032	0	0	17	8	2	0
20036	0	8	5	8	0	32
20040	39	8	9	0		
20044	5	9	4	12		
20048	8	29	29	-8		
20052	43	29	5	4		
20056	43	29	4	0		
20060	0	17	0	4	0	32
20064	0	6	0	5	0	32
20068	0	7	0	6	0	32
20072	3	10000				
20076	39	29	5	4		
20080	39	29	4	0		
20084	8	29	29	8		
20088	0	2	0	16	0	32
20092	2	5026				
20096	8	17	17	1		
20100	2	5006				
20104	0	16	0	2	0	32
20108	39	29	31	8		
20112	39	29	17	4		
20116	39	29	16	0		
20120	8	29	29	12		
20124	0	31	0	0	0	8

Machine Code for Next

Memory	op	rs	rt	rd	shamt	funct
				constant or address		
40000	8	29	29	-8		
40004	43	29	17	4		
40008	43	29	16	0		
40012	0	0	0	16	0	32
40016	0	0	0	17	0	32
40020	0	17	6	8	0	42
40024	4	8	0	9		
40028	0	0	17	8	2	0
40032	0	8	5	8	0	32
40036	39	8	9	0		
40040	5	9	4	2		
40044	8	17	17	1		
40048	2	10005				
40052	0	16	0	2	0	32
40056	39	29	17	4		
40060	39	29	16	0		
40064	8	29	29	8		
40068	0	31	0	0	0	8

High-Level Language Code

```
public int Search(int m, int[] A, int[] B, int len){
        int r = 0;
        for(int i = 0; i < len; i++){
                if (A[i] == m){
                        r = Next(i, B, len);
                        break;
                }
        }
        return r;
}

public int Next(int i, int[] B, int len){
        int k = 0;
        for(int j = 0; j < len; j++){
                if (B[j] == i){
                        k = j;
                        break;
                }
        }
        return k;
}
```

2) [25points] [Arithmetic for Computers] This problem covers 4-bit binary multiplication. Fill in the table for the Product, Multiplier and Multiplicand for each step. You need to provide the DESCRIPTION of the step being performed (shift left, shift right, add, no add). The value of M (Multiplicand) is 1101, Q (Multiplier)is initially 1100.

Product	Multiplicand	Multiplier	Description	Step
0000 0000	0000 1101	1100	Initial values	0
0000 0000	0000 1101	1100	No add	1
0000 0000	0001 1010	1100	Shift left M	2
0000 0000	0001 1010	0110	Shift right Q	3
0000 0000	0001 1010	0110	No add	4
0000 0000	0011 0100	0110	Shift left M	5
0000 0000	0011 0100	0011	Shift right Q	6
0011 0100	0011 0100	0011	Add	7
0011 0100	0110 1000	0011	Shift left M	8
0011 0100	0110 1000	0001	Shift right Q	9
1001 1100	0110 1000	0001	Add	10
1001 1100	1101 0000	0001	Shift left M	11
1001 1100	1101 0000	0000	Shift right Q	12
				13
				14
				15

3) [25 points] [Arithmetic for Computers] Do following sub-questions.

a) Encode the following numbers into the 32-bit IEEE 754 floating point format (single precision).

i) $0.0110101_{binary} = 1.10101 \times 2^{-2}$

Exponent = -2 + 127 = 125 = 1111101

0	0	1	1	1	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ii) $13.625_{decimal} = 1101.101_{binary} = 1.101101 \times 2^{3}$

Exponent = 3 + 127 = 130 = 10000010

0	1	0	0	0	0	0	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b) Perform the computation $1.01100_{binary} \times 2^{-4} + 0.10111_{binary} \times 2^{-1}$ by assuming number of digits allowed after binary point is 5. Before computation, normalized the numbers.

$1.01100 \times 2^{-4} + 1.01110 \times 2^{-2}$

$0.01011 \times 2^{-2} + 1.01110 \times 2^{-2} = 1.11001 \times 2^{-2}$

c) Perform the computation of $2.4375_{decimal} \times -0.1875_{decimal}$ in binary form (First convert the numbers into binary).

$2.4375_{decimal} = 10.0111_{binary} = 1.00111 \times 2^{1}$

$-0.1875_{decimal} = -0.0011_{binary} = -1.10000 \times 2^{-3}$

Exponent = 1-3 = -2

1.00111 x 1.10000 = 1.110101

-1.110101×2^{-2}