Tuesday, September 18, 2018

ECE366 Project 1

Yihua PU

Program I – “6^P mod 17”

Code:

```
# Project 1   Calculate the result of "6^P mod 17"
# Author: Yihua PU

.data
P: .word 1005
R: .word -1           # the result

even_result: .word 1,2,4,8,16,15,13,9
odd_result: .word 6,12,7,14,11,5,10,3
# the steps and ideas about how to
# caculate these numbers are on my scratch paper attached below.

.text
        lw $8, 0x2000($0)    # load the P into register
        andi $9,$8,1         #  get the lowest bit of P
        bne $9,$0,odd        #  P is even or odd ?
even:
        andi $8,$8,0xF       # "6^P mod 17" have 16-number-cycle. So P = P mod 16
        sll $8,$8,1          # relative word address = P*2
        addi $10,$0,0x2008   # the address of even_result array
        add $10,$10,$8       # the address of answer
        j then
odd:
        andi $8,$8,0xF       # "6^P mod 17" have 16-number-cycle. So P = P mod 16
        addi $8,$8,-1        # relative word address = (P-1)*2
        sll $8,$8,1
        addi $10,$0,0x2028   # the address of odd_result array
        add $10,$10,$8       # the address of answer
then:
        lw $11,($10)
        sw $11,0x2004($0)

```

If you wanna run and test it :

```
# Project 1   Calculate the result of "6^P mod 17"
# Author: Yihua PU

.data
P: .word 1005
R: .word -1           # the result

even_result: .word 1,2,4,8,16,15,13,9
odd_result: .word 6,12,7,14,11,5,10,3
# the steps and ideas about how to
# caculate these numbers are on my scratch paper attached below.

.text
       lw $8, 0x2000($0)    # load the P into register
       andi $9,$8,1         #  get the lowest bit of P
       bne $9,$0,odd        #  P is even or odd ?
even:
       andi $8,$8,0xF       # "6^P mod 17" have 16-number-cycle. So P = P mod 16
       sll $8,$8,1          # relative word address = P*2
       addi $10,$0,0x2008   # the address of even_result array
       add $10,$10,$8           # the address of answer
       j then
odd:
       andi $8,$8,0xF       # "6^P mod 17" have 16-number-cycle. So P = P mod 16
       addi $8,$8,-1        # relative word address = (P-1)*2
       sll $8,$8,1
       addi $10,$0,0x2028   # the address of odd_result array
       add $10,$10,$8       # the address of answer
then:
       lw $11,($10)
       sw $11,0x2004($0)
```

Tuesday, September 18, 2018

Questions :

1) Level 2 with low DIC

2)

P = 8:

Text Segment

Bkpt	Address	Code	Basic	Source		
	0x00000000	0x8c082000	lw $8,8192($0)	14:	lw $8, 0x2000($0)	# load the P into register
	0x00000004	0x31090001	andi $9,$8,1	15:	andi $9,$8,1	# get the lowest bit of P
	0x00000008	0x15200005	bne $9,$0,5	16:	bne $9,$0,odd	# P is even or odd ?
	0x0000000c	0x3108000f	andi $8,$8,15	18:	andi $8,$8,0xF	# "6^P mod 17" have 16-number-cycle. So P = P mod 16
	0x00000010	0x00084040	sll $8,$8,1	19:	sll $8,$8,1	# relative word address = P*2
	0x00000014	0x200a2008	addi $10,$0,8200	20:	addi $10,$0,0x2008	# the address of even_result array
	0x00000018	0x01485020	add $10,$10,$8	21:	add $10,$10,$8	# the address of answer
	0x0000001c	0x0800000d	j 0x00000034	22:	j then	
	0x00000020	0x3108000f	andi $8,$8,15	24:	andi $8,$8,0xF	# "6^P mod 17" have 16-number-cycle. So P = P mod 16
	0x00000024	0x2108ffff	addi $8,$8,-1	25:	addi $8,$8,-1	# relative word address = (P-1)*2
	0x00000028	0x00084040	sll $8,$8,1	26:	sll $8,$8,1	
	0x0000002c	0x200a2028	addi $10,$0,8232	27:	addi $10,$0,0x2028	# the address of odd_result array
	0x00000030	0x01485020	add $10,$10,$8	28:	add $10,$10,$8	# the address of answer
	0x00000034	0x8d4b0000	lw $11,0($10)	30:	lw $11,($10)	
	0x00000038	0xac0b2004	sw $11,8196($0)	31:	sw $11,0x2004($0)	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00002000	8	16	1	2	4	
0x00002020	13	9	6	12	7	
0x00002040	10	3	0	0	0	

Text Segment

Bkpt	Address	Code	Basic	Source		
	0x00000000	0x8c082000	lw $8,8192($0)	14:	lw $8, 0x2000($0)	# load the P into register
	0x00000004	0x31090001	andi $9,$8,1	15:	andi $9,$8,1	# get the lowest bit of P
	0x00000008	0x15200005	bne $9,$0,5	16:	bne $9,$0,odd	# P is even or odd ?
	0x0000000c	0x3108000f	andi $8,$8,15	18:	andi $8,$8,0xF	# "6^P mod 17" have 16-number-cycle. So P = P mod 1
	0x00000010	0x00084040	sll $8,$8,1	19:	sll $8,$8,1	# relative word address = P*2
	0x00000014	0x200a2008	addi $10,$0,8200	20:	addi $10,$0,0x2008	# the address of even_result array
	0x00000018	0x01485020	add $10,$10,$8	21:	add $10,$10,$8	# the address of answer
	0x0000001c	0x0800000d	j 0x00000034	22:	j then	
	0x00000020	0x3108000f	andi $8,$8,15	24:	andi $8,$8,0xF	# "6^P mod 17" have 16-number-cycle. So P = P mod 1
	0x00000024	0x2108ffff	addi $8,$8,-1	25:	addi $8,$8,-1	# relative word address = (P-1)*2
	0x00000028	0x00084040	sll $8,$8,1	26:	sll $8,$8,1	
	0x0000002c	0x200a2028	addi $10,$0,8232	27:	addi $10,$0,0x2028	# the address of odd_result array
	0x00000030	0x01485020	add $10,$10,$8	28:	add $10,$10,$8	# the address of answer
	0x00000034	0x8d4b0000	lw $11,0($10)	30:	lw $11,($10)	
	0x00000038	0xac0b2004	sw $11,8196($0)	31:	sw $11,0x2004($0)	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+1
0x00002000	201	11	1	2	4	
0x00002020	13	9	6	12	7	
0x00002040	10	3	0	0	0	

P = 201

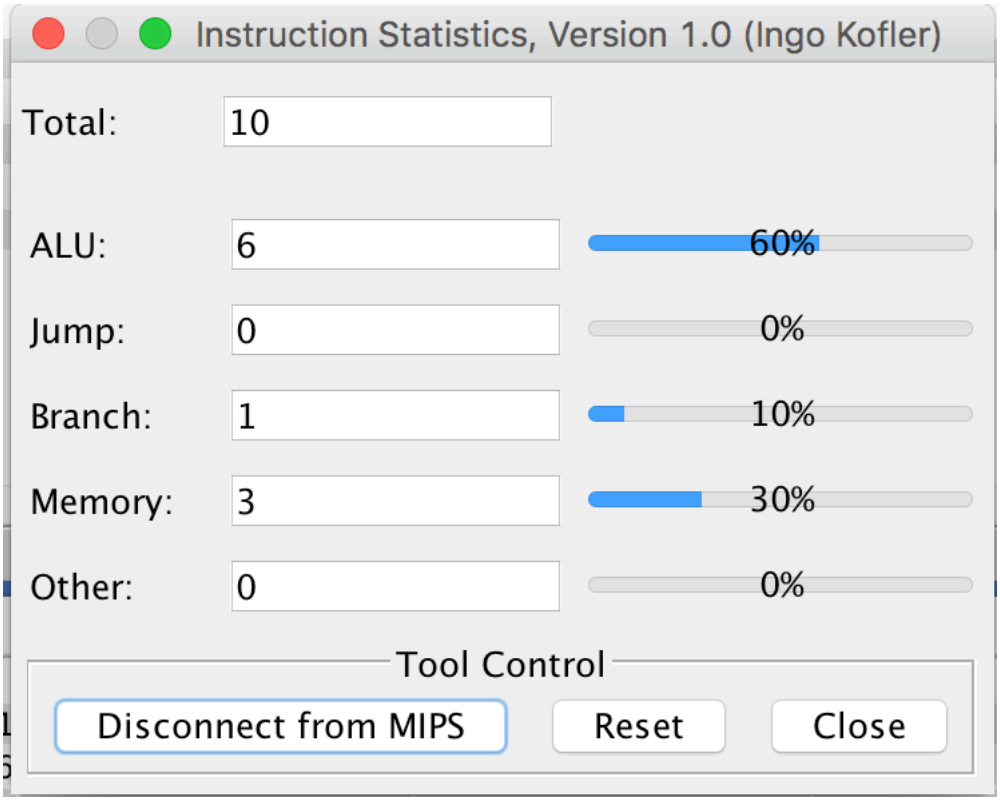
P = 1005:

Text Segment

Bkpt	Address	Code	Basic	Source		
	0x00000000	0x8c082000	lw $8,8192($0)	14:	lw $8, 0x2000($0)	# load the P into register
	0x00000004	0x31090001	andi $9,$8,1	15:	andi $9,$8,1	# get the lowest bit of P
	0x00000008	0x15200005	bne $9,$0,5	16:	bne $9,$0,odd	# P is even or odd ?
	0x0000000c	0x3108000f	andi $8,$8,15	18:	andi $8,$8,0xF	# "6^P mod 17" have 16-number-cycle. So P = P mod 16
	0x00000010	0x00084040	sll $8,$8,1	19:	sll $8,$8,1	# relative word address = P*2
	0x00000014	0x200a2008	addi $10,$0,8200	20:	addi $10,$0,0x2008	# the address of even_result array
	0x00000018	0x01485020	add $10,$10,$8	21:	add $10,$10,$8	# the address of answer
	0x0000001c	0x0800000d	j 0x00000034	22:	j then	
	0x00000020	0x3108000f	andi $8,$8,15	24:	andi $8,$8,0xF	# "6^P mod 17" have 16-number-cycle. So P = P mod 16
	0x00000024	0x2108ffff	addi $8,$8,-1	25:	addi $8,$8,-1	# relative word address = (P-1)*2
	0x00000028	0x00084040	sll $8,$8,1	26:	sll $8,$8,1	
	0x0000002c	0x200a2028	addi $10,$0,8232	27:	addi $10,$0,0x2028	# the address of odd_result array
	0x00000030	0x01485020	add $10,$10,$8	28:	add $10,$10,$8	# the address of answer
	0x00000034	0x8d4b0000	lw $11,0($10)	30:	lw $11,($10)	
	0x00000038	0xac0b2004	sw $11,8196($0)	31:	sw $11,0x2004($0)	

Data Segment

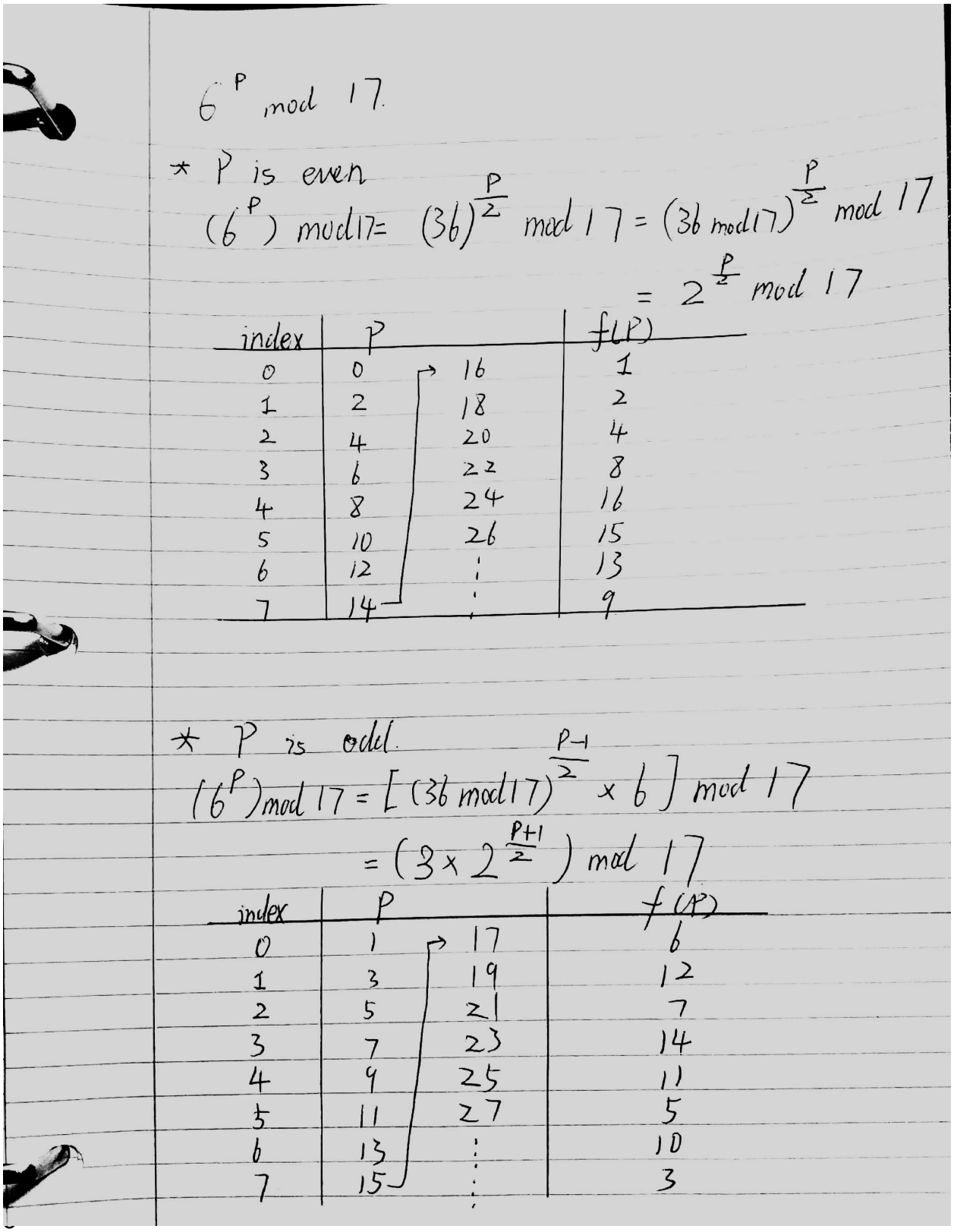
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00002000	1005	10	1	2	4	
0x00002020	13	9	6	12	7	
0x00002040	10	3	0	0	0	
0x00002060	0	0	0	0	0	



Tuesday, September 18, 2018

3)

I found that the value of expression “6^P mod 17” is periodic, and its cycle is 16. The calculation of its value can be divided into two situation depending on P is odd or even. Detailed procedures are attached below.



$6^P \bmod 17$

* P is even

$(6^P) \bmod 17 = (36)^{\frac{P}{2}} \bmod 17 = (36 \bmod 17)^{\frac{P}{2}} \bmod 17$

$= 2^{\frac{P}{2}} \bmod 17$

index	P		f(P)
0	0	16	1
1	2	18	2
2	4	20	4
3	6	22	8
4	8	24	16
5	10	26	15
6	12	⋮	13
7	14	⋮	9

* P is odd.

$(6^P) \bmod 17 = [(36 \bmod 17)^{\frac{P-1}{2}} \times 6] \bmod 17$

$= (3 \times 2^{\frac{P+1}{2}}) \bmod 17$

index	P		f(P)
0	1	17	6
1	3	19	12
2	5	21	7
3	7	23	14
4	9	25	11
5	11	27	5
6	13	⋮	10
7	15	⋮	3

4)

Instructions	Registers
lw, andi, bne, sll, addi, add, j, sw	$8,$9,$0,$10,$11

Program II – "Best Match Count"

Code:

```
.data
T: .word 12
best_matching_score: .word -1 # best score = ? within [0, 32]
best_matching_count: .word -1 # how many patterns achieve the best score?
Pattern_Array: .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20


.text
        lw $8, 0x2000($0)
        addi $10,$0,0x200C
        addi $16,$0,20    # exit traversal flag

        addi $17,$17,-1   # the greatest matching score
        addi $15,$15,0    # the greatest score counter

traversal:
        lw $13,0x0($10)
        xor $13,$13,$8

        # count 1's in a 32bit number
        xori $13,$13,0xFFFFFFFF
        srl $14,$13,1
        andi $13,$13,0x55555555   # imm used as filters to count 1's in a 32-bit binary number
        andi $14,$14,0x55555555
        addu $13,$13,$14

        srl $14,$13,2
        andi $13,$13,0x33333333
        andi $14,$14,0x33333333
        addu $13,$13,$14
```

```
31
32          srl $14,$13,4
33          andi $13,$13,0x07070707
34          andi $14,$14,0x07070707
35          addu $13,$13,$14
36
37          srl $14,$13,8
38          andi $13,$13,0x000F000F
39          andi $14,$14,0x000F000F
40          addu $13,$13,$14
41
42          srl $14,$13,16
43          andi $13,$13,0x0000001F
44          andi $14,$14,0x0000001F
45          addu $13,$13,$14          # the number of 1's is stored in $13
46
47  count_out:
48          beq $17,$13,equal       #if both of the two scores are equal
49          slt $18,$17,$13
50          beq $18,$0,pass         # if current score is greater than biggest score
51
52  greater:                        # if current score is greater than biggest score
53          add $17,$0,$13      # save the biggest score
54          addi $15,$0,1       # set the counter to 1
55          j pass
56  equal:
57          addi $15,$15,1
58  pass:
59          addi $10,$10,4    # prepare for next iteration
60
61          addi $16,$16,-1
62          beq $16,$0,trave_out
63          j traversal
64
65  trave_out:
66          sw $17,0x2004($0)  # save the greatest matching score
67          sw $15,0x2008($0)  # save the matching numbers
68
```

If you wanna run and test it :

```
.data
T: .word 12
best_matching_score: .word -1 # best score = ? within [0, 32]
best_matching_count: .word -1 # how many patterns achieve the best score?
Pattern_Array: .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18,
19, 20


.text
       lw $8, 0x2000($0)
       addi $10,$0,0x200C
       addi $16,$0,20    # exit traversal flag

       addi $17,$17,-1   # the greatest matching score
       addi $15,$15,0    # the greatest score counter

traversal:
       lw $13,0x0($10)
       xor $13,$13,$8

       # count 1's in a 32bit number
       xori $13,$13,0xFFFFFFFF
       srl $14,$13,1
       andi $13,$13,0x55555555   # imm used as filters to count 1's in a 32-bit
binary number
       andi $14,$14,0x55555555
       addu $13,$13,$14

       srl $14,$13,2
       andi $13,$13,0x33333333
       andi $14,$14,0x33333333
       addu $13,$13,$14

       srl $14,$13,4
       andi $13,$13,0x07070707
       andi $14,$14,0x07070707
       addu $13,$13,$14

       srl $14,$13,8
       andi $13,$13,0x000F000F
       andi $14,$14,0x000F000F
       addu $13,$13,$14
```

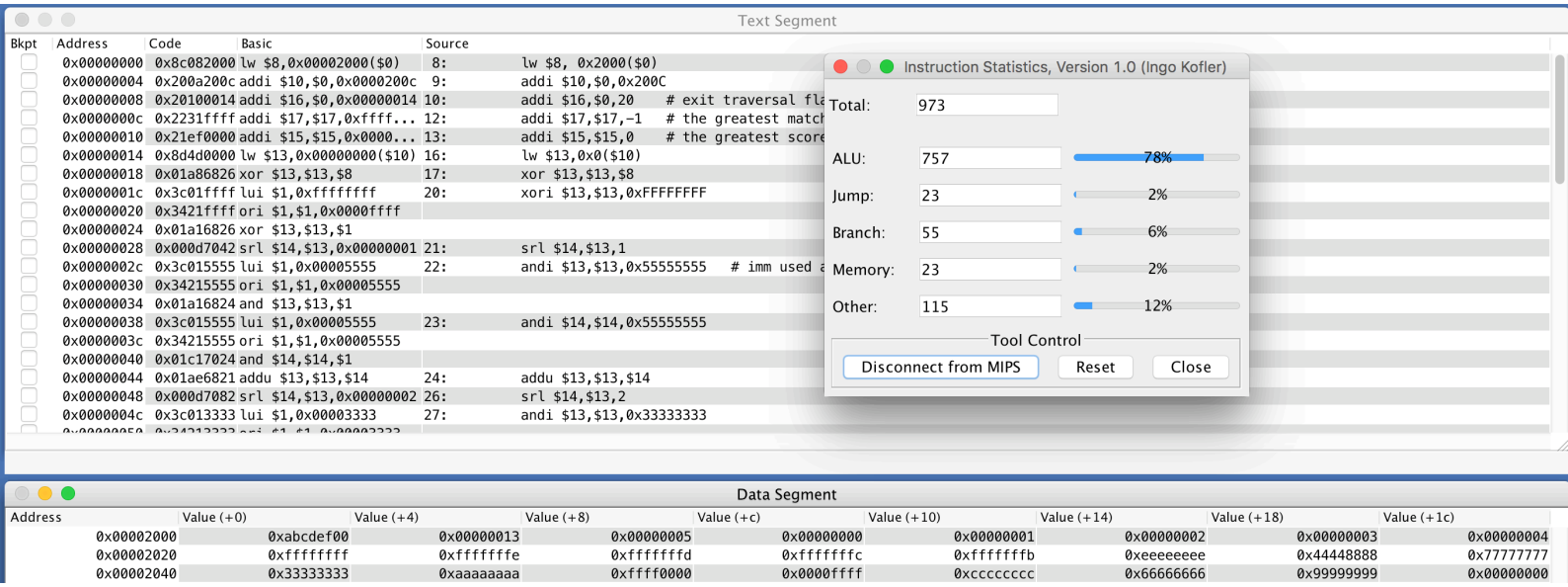
```
       srl $14,$13,16
       andi $13,$13,0x0000001F
       andi $14,$14,0x0000001F
       addu $13,$13,$14         # the number of 1's is stored in $13

count_out:
       beq $17,$13,equal   #if both of the two scores are equal
       slt $18,$17,$13
       beq $18,$0,pass         # if current score is greater than biggest score

greater:                   # if current score is greater than biggest score
       add $17,$0,$13     # save the biggest score
       addi $15,$0,1   # set the counter to 1
       j pass
equal:
       addi $15,$15,1
pass:
       addi $10,$10,4    # prepare for next iteration

       addi $16,$16,-1
       beq $16,$0,trave_out
       j traversal

trave_out:
       sw $17,0x2004($0)  # save the greatest matching score
       sw $15,0x2008($0)  # save the matching numbers
```

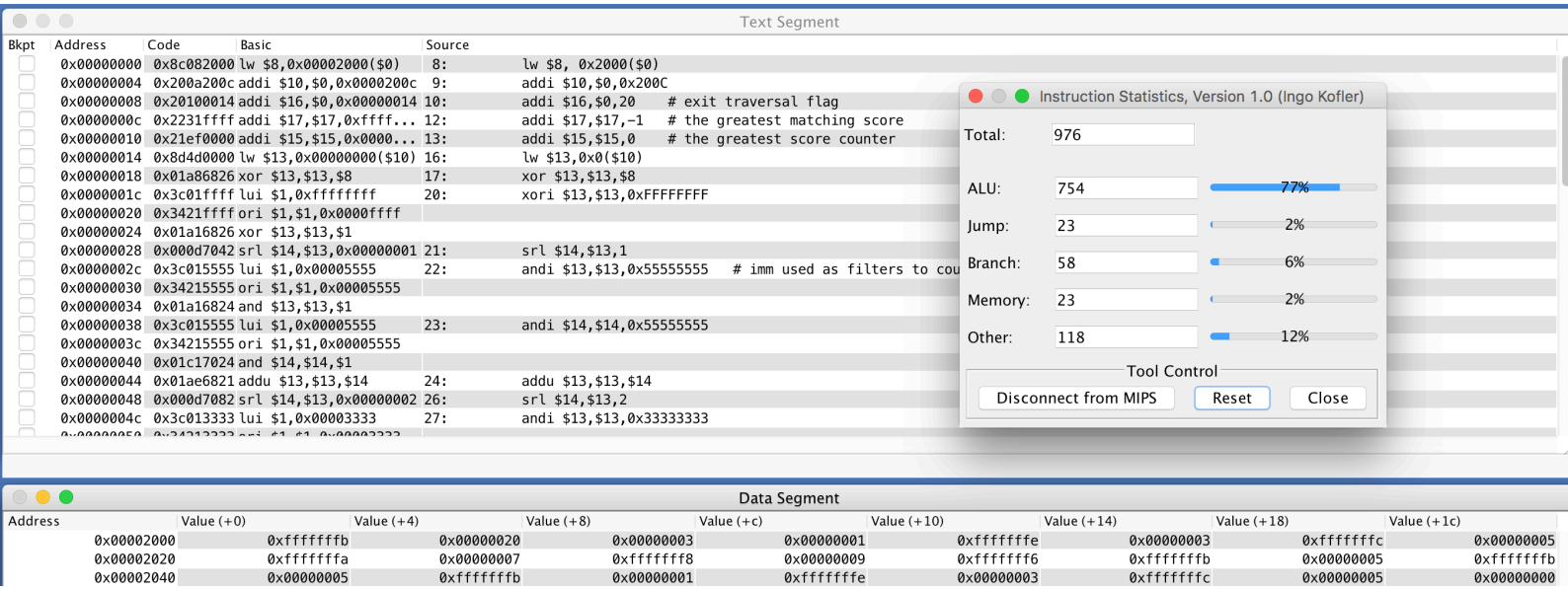
Questions :

5) level 2 with low DIC

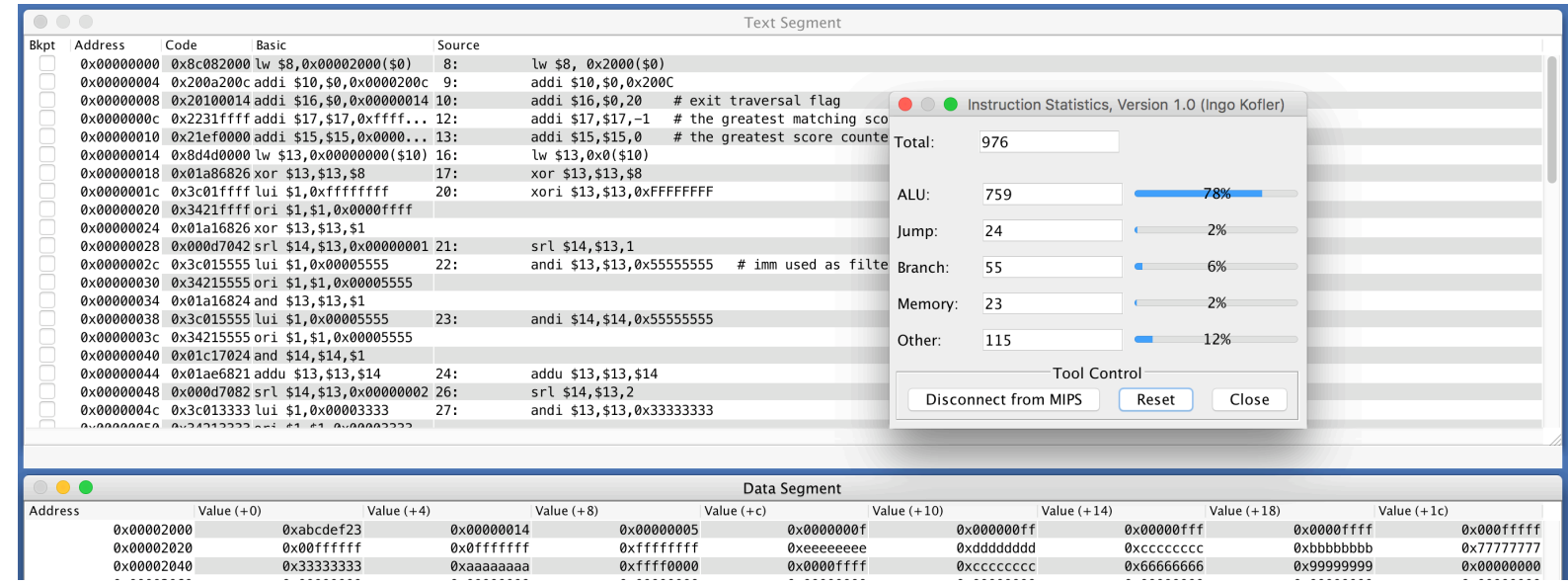
6)

Tuesday, September 18, 2018






```
T: .word 0xABCDEF23

Pattern_Array: .word 0xF, 0xFF, 0xFFF, 0xFFFF, 0xFFFFF, 0xFFFFFF, 0xFFFFFFF,
0xFFFFFFFF, 0xEEEEEEEE, 0xDDDDDDDD, 0xCCCCCCCC, 0xBBBBBBBB, 0x77777777,
0x33333333, 0xAAAAAAAA, 0xFFFF0000, 0xFFFF, 0xCCCCCCCC, 0x66666666,
0x99999999
```



7)

To find the number of same bits:

1. use XOR to find the same bits of two words, for the same bits we got 0, for the different bits we got 1.

2. use “result XOR 0xFFFFFFFF” to invert the result.

3. add the all the bits of that result words together, the value of that words is the number of those same bits.

To record the scores and count the number of best scores:

1. Initialize the count with 0, the best score with -1
2. For every given word, do the above-mentioned steps, get its score.
3. Compare the current score with the recorded best score, if both of them are equal, counter++; if current score is great than recorded best score, set counter to 0, set best score to current score.
4. When the traversal is done, store the score and counter as required.

8)

Instructions	Registers
`lw, sw, add, addi, addu, xor, xori, srl, andi, beq, slt, j`	`$0,$8,$9,$10,$11,$13,$14, $15,$16,$17,$18`

9) I can’t recall it clearly, maybe two nights. Writing this document indeed cost me a lot of time.

10)

For the program 1, I use something tricky—I do not do a lot of calculation. Instead, I use the periodicity to solve the problem. And I think maybe there are not too much space to optimize.

For the program 2, at first, I use a loop to calculate the sum of every bit of XOR result. The DIC is pretty high, about 4000.

To optimize this problem, I attempt to add counts in a tree pattern, and it make sense. The DIC is under 1000 now.

Talking about how to further optimize it, I don’t have a clear idea about that, maybe I can use a more efficient algorithm to calculate the Hamming weight?