Rafay Usmani
Anas Shalabi
Jim Palomo

Project 1 Report | Group 7

1. **Introduction**:

   Project 1 consisted of writing an assembly program in MIPS. The program is designed to have three parts:

   1. Generate an array “A” of size 100 where each index in the array contains a number. The number that is found each index of the array is determined by 3 given values: A, B, and C. The formula for obtaining the numbe ris (A * B) xor C where (A * B) is a “lo” 32 bit number. Then store each number in the range [0x20A0, 0x2230].

   2. Given the array “A,” determine the width of each number in the array (by loading in from each index of A). Obtain the width by finding the 1 closest (or at) MSB and the 1 closest (or at) LSB. Determine the length given these parameters (e.g. 0101 1010 → width of 6). Then store the array “W” into [0x22A0, 0x2430]. Since an index is made for each number of array A, W will also be of size 100.

   3. After generating and computing arrays A & W, produce an array H that stores the number of times a width has occurred ranging from [0, 32]. Store this into [0x2000, 0x2080]. This was done by loading from the H-array and incrementing the value stored within the H-array by 1. Then storing the incremented count back into the H-array.

Rafay Usmani
Anas Shalabi
Jim Palomo

Q&A:

a) Which parts (I, II, III, IV) does your program achieve?

Our program achieves part I, II, and III.

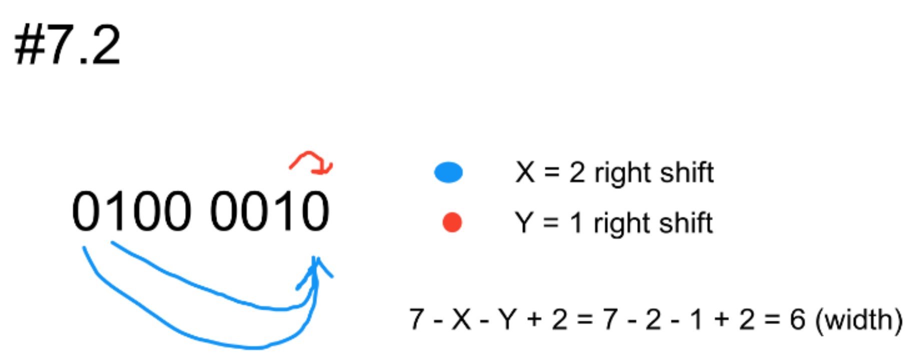
b) What was the most difficult part of this project?

The most difficult part of the project was Part 2. Part 2 was the most difficult because we had to draft different applications of what Part 2 should be. One of our early applications was to use left and right shifts however, using left shifts past b31 (bit 31) led to an overflow. Therefore, the bit would disappear after one more left shift. This was known after trying to apply the left shifting to Part 2 in MIPS.

We had to think of another way of accomplishing a left shift without using a left shift. Thus, we replicated using a left shift by only using right shifts. Therefore, our program uses two right shifts to traverse through each bit.

Moreover, after applying two loops where one replicated a left shift using only a right shift and a normal right shift, we had to figure out special cases. These special cases were known after figuring out what our algorithm can and can not do.

Here is a sample of what we achieved:



c) (if you worked individually) How many hours have you spent in total on this project?

N/A. Group work.

Rafay Usmani
Anas Shalabi
Jim Palomo

d) (if you worked in a group) Provide a rough breakdown of each member's help / contribution to what you are able submitting here (for example: 30% / 30% / 40%).

Name	Contribution (%)
Rafay Usmani	33.3
Anas Shalabi	33.3
Jim Palomo	33.3

Rafay Usmani
Anas Shalabi
Jim Palomo

2. Program functionality
   Run your program for each of the following configurations:
   a. A = 5, B = 2, C = 0x00000000
   b. A = 5, B = 2, C = 0x0000FFFF
   c. A = -3, B = -7, C = 0xFFFFFFFF
   d. Your selection of A, B, C that best "spreads" out the histogram of H0 to H32.

   Provide MARS screenshots of your program's results for each of the above configurations, including:
   - Final results of all the arrays
     - Show MARs' data memory content at regions such as:
       - 0x2000, 0x2020, 0x2120, etc
   - Dynamic Instruction count of completing your program
     - Use MARS Tools -> Instruction Statistics

A1 – A100:	at M[0x20A0, 0x20A4, …]	[8352, 8752]	[0x20A0, 0x2230]
W1 – W100:	at M[0x22A0, 0x22A4, …]	[8864, 9264]	[0x22A0, 0x2430]
H0 – H32:	at M[0x2000, 0x2004, …]	[8192, 8320]	[0x2000, 0x2080]

Rafay Usmani
Anas Shalabi
Jim Palomo

a. A = 5, B = 2, C = 0x00000000

A-array:

8352	10	20	40	80	160	320	640	1280
8384	2560	5120	10240	20480	40960	81920	163840	327680
8416	655360	1310720	2621440	5242880	10485760	20971520	41943040	83886080
8448	167772160	335544320	671088640	1342177280	-1610612736	1073741824	-2147483648	0
8480	0	0	0	0	0	0	0	0
8512	0	0	0	0	0	0	0	0
8544	0	0	0	0	0	0	0	0
8576	0	0	0	0	0	0	0	0

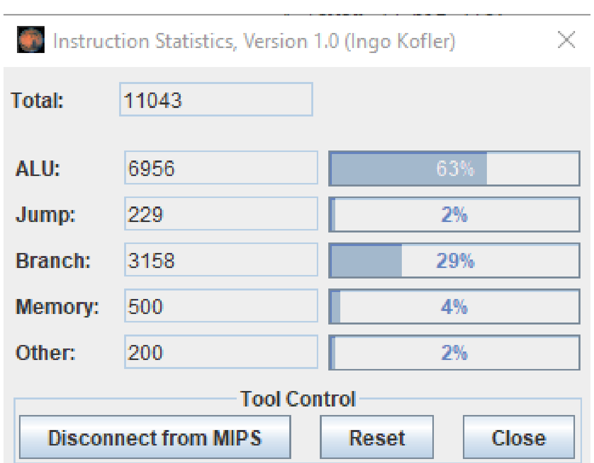
8448	167772160	335544320	671088640	1342177280	-1610612736	1073741824	-2147483648	0
8480	0	0	0	0	0	0	0	0
8512	0	0	0	0	0	0	0	0
8544	0	0	0	0	0	0	0	0
8576	0	0	0	0	0	0	0	0
8608	0	0	0	0	0	0	0	0
8640	0	0	0	0	0	0	0	0
8672	0	0	0	0	0	0	0	0
8704	0	0	0	0	0	0	0	0
8736	0	0	0	0	0	0	0	0

W-array:

8864	3	3	3	3	3	3	3	3
8896	3	3	3	3	3	3	3	3
8928	3	3	3	3	3	3	3	3
8960	3	3	3	3	3	1	1	0
8992	0	0	0	0	0	0	0	0
9024	0	0	0	0	0	0	0	0
9056	0	0	0	0	0	0	0	0
9088	0	0	0	0	0	0	0	0
9120	0	0	0	0	0	0	0	0
9152	0	0	0	0	0	0	0	0
9184	0	0	0	0	0	0	0	0
9216	0	0	0	0	0	0	0	0
9248	0	0	0	0	0	0	0	0

H-array:

8192	69	2	0	29	0	0	0	0
8224	0	0	0	0	0	0	0	0
8256	0	0	0	0	0	0	0	0
8288	0	0	0	0	0	0	0	0
8320	0	0	0	0	0	0	0	0



Rafay Usmani
Anas Shalabi
Jim Palomo

b. A = 5, B = 2, C = 0x0000FFFF

A-array:

8352	65525	65557	196565	327765	720725	1376597	2817365	5571925
8384	11203925	22353237	44750165	89478485	178935125	357848405	715674965	1431328085
8416	-1432332971	1430279509	-1434430123	1426085205	-1442818731	1409307989	-1476373163	1342199125
8448	-1610590891	1073763669	-2147461803	21845	21845	21845	21845	21845
8480	21845	21845	21845	21845	21845	21845	21845	21845
8512	21845	21845	21845	21845	21845	21845	21845	21845
8544	21845	21845	21845	21845	21845	21845	21845	21845
8576	21845	21845	21845	21845	21845	21845	21845	21845

8448	-1610590891	1073763669	-2147461803	21845	21845	21845	21845	21845
8480	21845	21845	21845	21845	21845	21845	21845	21845
8512	21845	21845	21845	21845	21845	21845	21845	21845
8544	21845	21845	21845	21845	21845	21845	21845	21845
8576	21845	21845	21845	21845	21845	21845	21845	21845
8608	21845	21845	21845	21845	21845	21845	21845	21845
8640	21845	21845	21845	21845	21845	21845	21845	21845
8672	21845	21845	21845	21845	21845	21845	21845	21845
8704	21845	21845	21845	21845	21845	21845	21845	21845
8736	21845	21845	21845	21845	0	0	0	0

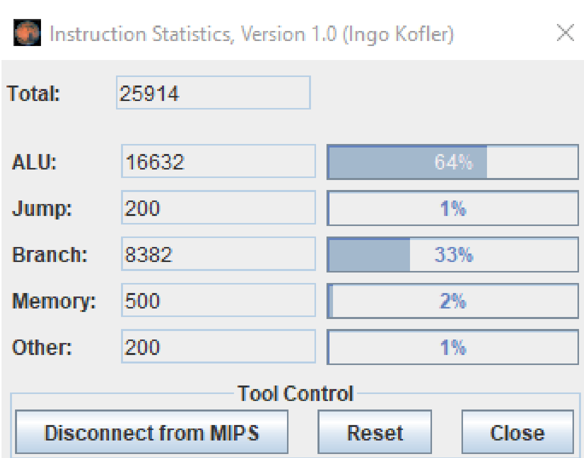
W- array

8864	16	17	18	19	20	21	22	23
8896	24	25	26	27	28	29	30	31
8928	32	31	32	31	32	31	32	31
8960	32	31	32	15	15	15	15	15
8992	15	15	15	15	15	15	15	15
9024	15	15	15	15	15	15	15	15
9056	15	15	15	15	15	15	15	15
9088	15	15	15	15	15	15	15	15

9120	15	15	15	15	15	15	15	15
9152	15	15	15	15	15	15	15	15
9184	15	15	15	15	15	15	15	15
9216	15	15	15	15	15	15	15	15
9248	15	15	15	15	0	0	0	0

H-array:

8192	0	0	0	0	0	0	0	0
8224	0	0	0	0	0	0	0	73
8256	1	1	1	1	1	1	1	1
8288	1	1	1	1	1	1	1	6
8320	6	0	0	0	0	0	0	0



Rafay Usmani
Anas Shalabi
Jim Palomo

c. A = -3, B = -7, C = 0xFFFFFFFF

A-array:

8352	-22	-155	-1086	-7603	-53222	-372555	-2607886	-18255203
8384	-127786422	-894504955	-1966567390	-881069843	-1872521606	-222749355	-1559245486	1970183485
8416	906382506	2049710245	1463069826	1651554189	-1324022566	-678223371	-452596302	1126793181
8448	-702382326	-621708987	-56995614	-398969299	1502182202	1925340821	592483858	-147580291
8480	-1033062038	1358500325	919567682	2142006477	2109143450	1879102261	268813938	1881697565
8512	286981066	2008867461	1177170338	-349742227	1846771706	42500053	297500370	2082502589
8544	1692616234	-1036588251	1333816834	746783245	932515418	-2062326667	-1551384782	2025208413
8576	1291557002	450964421	-1138216350	622420141	61973690	433815829	-1258256494	-217860867

8576	1291557002	450964421	-1138216350	622420141	61973690	433815829	-1258256494	-217860867
8608	-1525026070	-2085247899	-1711833406	902068045	2019509018	1251661237	171694066	1201858461
8640	-176925366	-1238477563	-79408350	-555858451	403958138	-1467260331	-1680887726	1118687805
8672	-759119958	-1018872411	1457827714	1614859405	-1580886054	1818699509	-154005326	-1078037283
8704	1043673610	-1284219323	-399600670	1497762605	1894403642	375923605	-1663502062	1240387453
8736	92777578	649443045	251134018	1757938125	0	0	0	0

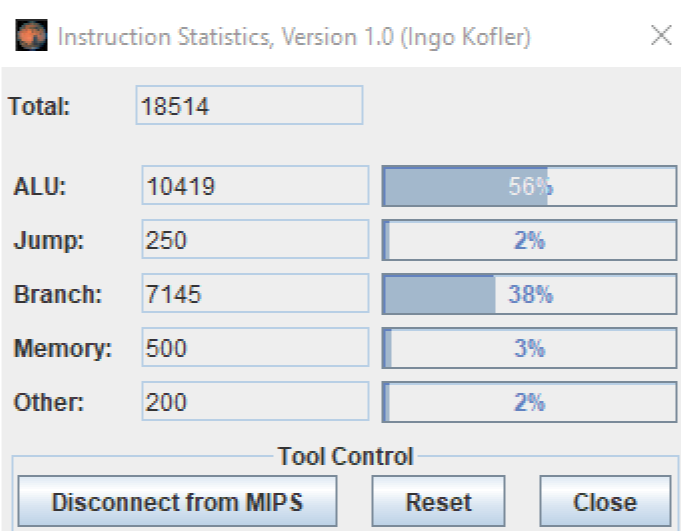
W-array:

8864	31	32	31	32	31	32	31	32
8896	31	32	31	32	31	32	31	31
8928	29	31	30	31	31	32	31	31
8960	31	32	31	32	30	31	29	32
8992	31	31	29	31	30	31	28	31
9024	28	31	30	32	30	26	28	31
9056	30	32	30	30	29	32	31	31
9088	30	29	31	30	25	29	31	32

9120	31	32	31	30	30	31	27	31
9152	31	32	31	32	28	32	31	31
9184	31	32	30	31	31	31	31	32
9216	29	32	31	31	30	29	31	31
9248	26	30	27	31	0	0	0	0

H-array:

8192	0	0	0	0	0	0	0	0
8224	0	0	0	0	0	0	0	0
8256	0	0	0	0	0	0	0	0
8288	0	1	2	2	4	8	15	46
8320	22	0	0	0	0	0	0	0



Rafay Usmani
Anas Shalabi
Jim Palomo

d. A = 8, B = -5, C = 0xFFFFFFFF

A-array:

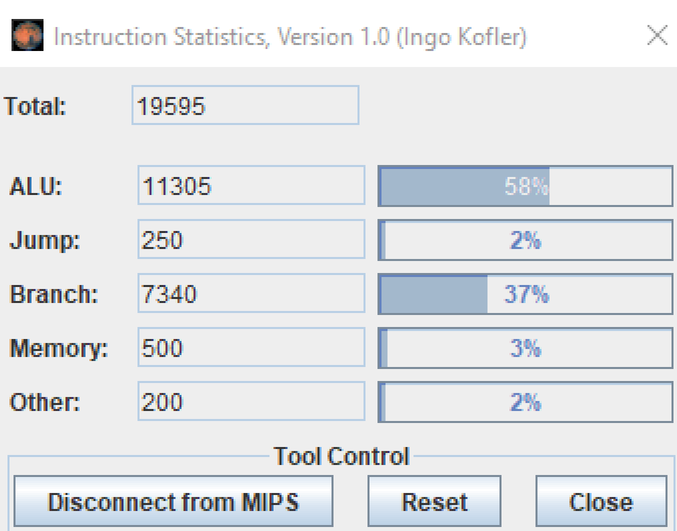
8352	39	194	969	4844	24219	121094	605469	3027344
8384	15136719	75683594	378417969	1892089844	870514627	57605838	288029189	1440145944
8416	-1389204873	1643910226	-370383463	-1851917316	-669651989	946707350	438569453	-2102120032
8448	-1920665569	-1013393254	-771998975	434972420	-2120105197	-2010591394	-1463022379	1274822696
8480	2079146183	1805796322	439047017	-2099732212	-1908726469	-953697754	-473521475	1927359920
8512	1046865007	939357738	401821393	2009106964	1455600227	-1311933458	2030267301	1561401912
8544	-782925033	380342130	1901710649	918618652	298125963	1490629814	-1136785523	-1388960320
8576	1645132991	-364269638	-1821348191	-516806364	1710935475	-35257218	-176286091	-881430456
8608	-112184985	-560924926	1490342665	-1138221268	-1396139045	1609239366	-543737763	1576278480
8640	-708542193	752256330	-533685647	1626539060	-457239293	2008770830	1453919557	-1320336808
8672	1988250551	1351318162	-1833343783	-576784324	1411045675	-1534706218	916403501	287050208
8704	1435251039	-1413679398	1521537601	-982246588	-616265645	1213639070	1773228053	276205672
8736	1381028359	-1684792798	165970601	829853004	0	0	0	0

W-array:

8864	6	7	10	11	15	16	20	18
8896	24	26	29	29	30	25	29	28
8928	32	30	32	30	32	29	29	27
8960	32	31	32	27	32	31	32	28
8992	31	30	29	30	32	31	32	27
9024	30	29	29	29	31	31	31	28
9056	32	28	31	28	29	30	32	26
9088	31	31	32	30	31	31	32	29
9120	32	31	31	30	32	30	32	27
9152	32	29	32	29	32	30	31	29
9184	31	30	32	30	31	31	30	24
9216	31	31	31	30	32	30	31	26
9248	31	31	28	28	0	0	0	0

H-array:

8192	0	0	0	0	0	0	1	1
8224	0	0	1	1	0	0	0	1
8256	1	0	1	0	1	0	0	0
8288	2	1	3	4	7	14	16	24
8320	21	0	0	0	0	0	0	0



Rafay Usmani
Anas Shalabi
Jim Palomo

3. Appendix

Include your MIPS assembly code here (-5 if missing)

```
#################################################################
# Project 1: Mips programming with MARS                        #
# 1) Generating an array of numbers A1 - A100                  #
# 2) Generating width array W1-W100                       #
# 3) Generating the histogram array of the widths H0 - H32  #
# Collaboration by Rafay Usmani, Anas Shalabi, and Jim Palomo     #
#################################################################

# Given
---------------------------------------------------------------------
---------------------------------------
addi $8, $0, 5                     # A = 5
addi $9, $0, -6               # B = -6
lui $10, 0xCD                 # C = 0x00CD0000
ori $10, $10, 0x1234               # C = 0x00CD1234

#
/////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////
# Part 1   Generating an array of numbers A1 - A100
                      /////
#
/////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////

# Start here
---------------------------------------------------------------------
-----------------------------------
addi $11, $0, 8352                 # $11 = 8352
addi $12, $0, 8752                 # $12 = 8524; last data segment for
part 1 will be 8352 + 400 = 8752
                           # note that 400 came from 100 (size of
array) * 4 (bytes per data segment).
addi $15, $0, 1               # $15 = 1
```

Rafay Usmani
Anas Shalabi
Jim Palomo

```
                      # $25 = 0

Loop:
mul $8, $8, $9                     # A(n) = A(n) * B [n = each loop
iteration]
mfhi $13              # $13 = hi
bne $25, $13, overflow             # if $25 = $13 [1=1] then there is
an overflow so branch
xor $8, $8, $10            # $8 = (A * B) xor C
j updateMemory             # branch to updateMemory

overflow:
mflo $8                    # lower 32 bit (lo)
xor $8, $8, $10            # $8 = (A * B) xor C
j updateMemory             # branch to updateMemory

updateMemory:
sw $8, 0($11)              # DM[$11] = $8
addi $11, $11, 4           # update memory location (+4) to form
array
bne $11, $12, Loop                 # stop when reached 100 element in
Part 1 array

#
/////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////
# Part 2   Generating width array W1-W100
                      /////
#
/////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////
addi $13, $0, 8328         # $13 = hold address for current 32 bit
addi $14, $0, 31      # $14 = 31 (for first loop, holds far left 1
bit position)
addi $15, $0, 1            # $15 = 1 (checks)

                      # $9 = temporary counter
```

```
                      # $10 = hold current 32 bit
                      # $11 = used to check bit before loops (MSB /
LSB)hold scanned bit / temp
                      # $12 = hold width
                      # $24 = 0 [counter for first right shifts]
                      # $25 = 0 [counter for second right shifts]


addi $1, $0, 8864           # $1 = 8864     DM[W] start
addi $2, $0, 9264           # $2 = 9264     DM[W] end
addi $8, $0, 8352           # $8 = 8352     DM[A] start
sub $9, $9, $9              # reset $9
addi $7, $0, 8192           # $7 = 8192     DM[H] start
addi $4, $0, 4              # $4 = 4

# Start Looping here after implementation
Loop_Part2_3:
lw $3, 0($8)                # $3 = DM[$8] = DM[8352] (start of
A-array)

addi $10, $3, 0             # reset $10 to original 32 bit

# Check for zero case [Special Case #1]
----------------------------------------------------------------------
--------
beq $10, $0, sp_case1

# Check for only 1 one [Special Case #2]
----------------------------------------------------------------------
-------
addi $10, $3, 0             # reset $10 to original 32 bit

lui $24, 0x8                # $24 = 0x00008000
sll $24, $24, 12      # $24 = 0x80000000 = 10000...0

addi $25, $0, 2             # $25 = 2
```

```
                      # check 0x80000000; signed so we have to check
for negative

beq $10, $24, sp_case2          # check if $8 (given) is $24
(0x80000000 = -2147483648)
srl $24, $24, 1

case_two_loop:
beq $10, $24, sp_case2          # check if $8 (given) is $24 (where
$24 is a 2^n value)
srl $24, $24, 1
addi $9, $9, 1             # temporary counter
bne $9, $14, case_two_loop # once $9 gets to 31 (iterations) stop

# Reset $9 = 0, $15 = 1, $24 = 0, $25 = 0
---------------------------------------------------------------------
------
sub $9, $9, $9             # reset $9 to 0
addi $15, $0, 1            # reset $15 to 1 ($15 = $15 + 2 = -1 + 2 =
1)
sub $24, $24, $24          # reset $24 to 0
sub $25, $25, $25          # reset $24 to 0

# Check for 1 @ MSB & LSB for original [Special Case 3]
-------------------------------------------------------------
lui $9, 0x8000             # $8 = 0x80000000
ori $9, $9, 1              # $8 = 0x80000001 = 1000...01
beq $9, $8, sp_case3

# Reset $9
---------------------------------------------------------------------
------------------------------------
sub $9, $9, $9             # reset $9 = 0

# Far_Left (1st Right Shifting Loop)
---------------------------------------------------------------------
-----------
```

Rafay Usmani
Anas Shalabi
Jim Palomo

```
Far_Left:
addi $10, $3, 0             # reset $10 to original 32 bit
srlv $10, $10, $14          # set [31 - # iterations] to LSB
addi $24, $24, 1       # $24 = counter for # of shifts
andi $10, $10, 1       # check shifted [MSB - # iterations] if it is a
1 or 0 by zero extend
                       # (e.g. 01001011 --> 00000001)
addi $14, $14, -1           # $14 = 31 - # iterations (set up for next
iteration)
bne $10, $15, Far_Left           # branch until 1 is found

# Check for 1 at LSB before Far_Right
----------------------------------------------------------------------
----------


addi $10, $3, 0             # reset $10 to original 32 bit
andi $11, $10, 1       # check LSB to see if shifting is needed
beq $11, $15, skip_right

# Far_Right (2nd Right Shifting Loop)
----------------------------------------------------------------------
----------


addi $10, $3, 0             # reset $10 to original 32 bit
Far_Right:
srl $10, $10, 1             # right shift data
andi $11, $10, 1       # $11 = stores LSB of $10
addi $25, $25, 1       # increment second right shift counter
bne $11, $15, Far_Right          # branch if 1 is not found by right
shifting
j past_skip_right_cond

skip_right:
```

```
addi $25, $0, 0             # set $25 to 0 since no shifts

past_skip_right_cond:

# Use Equation Here [Width = 31 - X - Y + 2]
---------------------------------------------------------------------
---
sub $14, $14, $14           # reset $14 to 0
addi $14, $14, 31           # $14 = 31
sub $14, $14, $24           # 31 - $24
sub $14, $14, $25           # 31 - $24 - $25
addi $12, $14, 2      # 31 - $24 - $25 + 2 --> WIDTH

j next_set

sp_case1:             # [Special Case: 1]
addi $12, $0, 0             # since the 32 bit totals to 0, then there
are no 1 bits so width is 0
j next_set            # jump for next iteration

sp_case2:             # [Special Cases: 2, 4, 5]
addi $12, $0, 1             # set width to 1 since there is only 1 bit
in the 32 bit original
j next_set            # jump for next iteration

sp_case3:             # [Special Case 3]
addi $12, $0, 32      # set width to 32 since there is only two 1's @
MSB and LSB
j next_set            # jump for next iteration

next_set:

# Set up next loop here
addi $14, $0, 31      # reset $14 to 31
sw $12, 0($1)               # store $12
addi $1, $1, 4              # update memory address by +4 for next
iteration
```

Rafay Usmani
Anas Shalabi
Jim Palomo

```
addi $8, $8, 4              # update memory address by +4 for next
iteration

#
////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////
# Part 3   Generating the histogram array of the widths H0 - H32
                            /////
#
////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////

mul $5, $12, $4             # $5 (offset) = $12 * $4 = Width * 4 (data
memory represented ending in 4's)
add $7, $7, $5              # add base + offset
lw $6 0($7)                 # load current count of width
addi $6, $6, 1              # increment value from DM[width]
sw $6 0($7)                 # store incremented value from DM[width]
addi $7, $0, 8192           # reset $7 [start of H]
bne $1, $2, Loop_Part2_3    # loop until all elements in W array are
accounted for
```