Homework #7 SOLUTION

1. Hand assemble the following code. Start instructions at address 0x0040 894C

```
*** original code:
    li  $t3, -47        # -47: 0xFFFFFFD1
    sll $v0, $t1, 3
    sw  $t8, -68($a2)   # -68: 0xFFBC

*** step #1: expand macro instructions:
    lui $at, 0xFFFF
    ori $v0, $at, 0xFFD1
    sll $v0, $t1, 3
    sw  $t8, -68($a2)

*** step #2: convert register names to register numbers:
    lui $1, 0xFFFF
    ori $2, $1, 0xFFD1
    sll $2, $9, 3
    sw  $24, -68($6)

*** step #3: align labels with assembly code:
    lui $1, 0xFFFF
    ori $2, $1, 0xFFD1
    sll $2, $9, 3
    sw  $24, -68($6)

*** step #4: convert labels to addresses:
    lui $1, 0xFFFF          <-- 0x0040 894C
    ori $2, $1, 0xFFD1      <-- 0x0040 8950
    sll $2, $9, 3           <-- 0x0040 8954
    sw  $24, -68($6)        <-- 0x0040 8958

*** step 5: calculate branch offsets:
    lui $1, 0xFFFF          <-- 0x0040 894C
    ori $2, $1, 0xFFD1      <-- 0x0040 8950
    sll $2, $9, 3           <-- 0x0040 8954
    sw  $24, -68($6)        <-- 0x0040 8958

*** step 6: convert instructions to machine code:
    lui $1, 0xFFFF          <-- 0x0040 894C
    0011 11-- ---t tttt iiii iiii iiii iiii
    0011 1100 0000 0001 1111 1111 1111 1111

    ori $2, $1, 0xFFD1      <-- 0x0040 8950
    0011 01ss ssst tttt iiii iiii iiii iiii
    0011 0100 0010 0010 1111 1111 1101 0001

    sll $2, $9, 3           <-- 0x0040 8954
    0000 0000 000t tttt dddd diii ii00 0000
    0000 0000 0000 1001 0001 0000 1100 0000

    sw  $24, -68($6)        <-- 0x0040 8958
    1010 11ss ssst tttt iiii iiii iiii iiii
    1010 1100 1101 1000 1111 1111 1011 1100
```

Answer:

```
    | address  |    instruction                              |
    ----------------------------------------------------------
    0x0040 894C: 0011 1100 0000 0001 1111 1111 1111 1111
    0x0040 8950: 0011 0100 0010 0010 1111 1111 1101 0001
    0x0040 8954: 0000 0000 0000 1001 0001 0000 1100 0000
    0x0040 8958: 1010 1100 1101 1000 1111 1111 1011 1100
```

2. To hand assemble this code you have to first convert the macros to actual instructions. Start instructions at address 0x0040 BC0C and the data start at address 0x1000 41AC.

```
*** original code:
            .data
value:   .word   -57         # 0xFFFFFFC7
```

```
            .text
    li $t4, -1              # initialize total, 0xFFFFFFFF
loop:
    blez $t2, loop_exit

    lw $t5, 0($t2)          # load values from matrices
    lw $t6, 0($t3)

    mul $t7, $t5, $t6
    add $t4, $t4, $t7

    addi $t1, $t1, -1       # decrement counter
    addiu $t2, $t2, 4       # adj mat1 ptr
    addu $t7, $t7, $t4      # adj mat2 ptr by offset

    b   loop

loop_exit:
    sw $t4, 20($sp)         # load return value to system stack


*** step #1: expand macro instructions:
            .data
value:  .word   -57         # 0xFFFFFFC7
            .text
    lui $at, 0xFFFF         # initialize total, 0xFFFFFFFF
    ori $t4, $at, 0xFFFF
loop:
    blez $t2, loop_exit

    lw $t5, 0($t2)          # load values from matrices
    lw $t6, 0($t3)

    mult $t5, $t6
    mflo $t7
    add $t4, $t4, $t7

    addi $t1, $t1, -1       # decrement counter
    addiu $t2, $t2, 4       # adj mat1 ptr
    addu $t7, $t7, $t4      # adj mat2 ptr by offset

    bgez $0, loop

loop_exit:
    sw $t4, 20($sp)         # load return value to system stack


*** step #2: convert register names to register numbers:
            .data
value:  .word   -57         # 0xFFFFFFC7
            .text
    lui $1, 0xFFFF          # initialize total, 0xFFFFFFFF
    ori $12, $1, 0xFFFF
loop:
    blez $10, loop_exit

    lw $13, 0($10)          # load values from matrices
    lw $14, 0($11)

    mult $13, $14
    mflo $15
    add $12, $12, $15

    addi $9, $9, -1         # decrement counter
    addiu $10, $10, 4       # adj mat1 ptr
    addu $15, $15, $12      # adj mat2 ptr by offset

    bgez $0, loop

loop_exit:
```

```
    sw $12, 20($19)         # load return value to system stack


*** step #3: align labels with assembly code:
            .data
value:  .word   -57         # 0xFFFFFFC7
            .text
    lui $1, 0xFFFF          # initialize total, 0xFFFFFFFF
    ori $12, $1, 0xFFFF
loop: blez $10, loop_exit

    lw $13, 0($10)          # load values from matrices
    lw $14, 0($11)

    mult $13, $14
    mflo $15
    add $12, $12, $15

    addi $9, $9, -1         # decrement counter
    addiu $10, $10, 4       # adj mat1 ptr
    addu $15, $15, $12      # adj mat2 ptr by offset

    bgez $0, loop

loop_exit: sw $12, 20($19)  # load return value to system stack


*** step #4: convert labels to addresses:
            .data
value:  .word   -57         # 0xFFFFFFC7                        | 0x1000 41AC
            .text
    lui $1, 0xFFFF          # initialize total, 0xFFFFFFFF      | 0x0040 BC0C
    ori $12, $1, 0xFFFF                                         | 0x0040 BC10
loop: blez $10, loop_exit                                       | 0x0040 BC14

    lw $13, 0($10)          # load values from matrices         | 0x0040 BC18
    lw $14, 0($11)                                              | 0x0040 BC1C

    mult $13, $14                                               | 0x0040 BC20
    mflo $15                                                    | 0x0040 BC24
    add $12, $12, $15                                           | 0x0040 BC28

    addi $9, $9, -1         # decrement counter                 | 0x0040 BC2C
    addiu $10, $10, 4       # adj mat1 ptr                      | 0x0040 BC30
    addu $15, $15, $12      # adj mat2 ptr by offset            | 0x0040 BC34

    bgez $0, loop                                               | 0x0040 BC38

loop_exit: sw $12, 20($19)   # load return value                | 0x0040 BC3C
                             # to system stack

*** step 5: calculate branch offsets:
            .data
value:  .word   -57         # 0xFFFFFFC7                        | 0x1000 41AC
            .text
    lui $1, 0xFFFF          # initialize total, 0xFFFFFFFF      | 0x0040 BC0C
    ori $12, $1, 0xFFFF                                         | 0x0040 BC10
loop: blez $10, loop_exit                                       | 0x0040 BC14
#   offset = distance - 1 => 10 - 1 = 9


    lw $13, 0($10)          # load values from matrices         | 0x0040 BC18
    lw $14, 0($11)                                              | 0x0040 BC1C

    mult $13, $14                                               | 0x0040 BC20
    mflo $15                                                    | 0x0040 BC24
    add $12, $12, $15                                           | 0x0040 BC28

    addi $9, $9, -1         # decrement counter                 | 0x0040 BC2C
    addiu $10, $10, 4       # adj mat1 ptr                      | 0x0040 BC30
```

```
    addu $15, $15, $12      # adj mat2 ptr by offset            | 0x0040 BC34

    bgez $0, loop                                               | 0x0040 BC38
#   offset = distance - 1 => -9 - 1 = -10

loop_exit: sw $12, 20($19)  # load return value                 | 0x0040 BC3C
                            # to system stack


*** step 6: convert instructions to machine code:
            .data
value:  .word   -57         # 0xFFFFFFC7                        | 0x1000 41AC
            .text
    lui $1, 0xFFFF          # initialize total, 0xFFFFFFFF      | 0x0040 BC0C
    0011 1100 000t tttt iiii iiii iiii iiii
    0011 1100 0000 0001 1111 1111 1111 1111

    ori $12, $1, 0xFFFF                                         | 0x0040 BC10
    0011 01ss ssst tttt iiii iiii iiii iiii
    0011 0100 0010 1100 1111 1111 1111 1111
loop: blez $10, loop_exit                                       | 0x0040 BC14
#   offset = distance - 1 => 10 - 1 = 9
    0001 10ss sss0 0000 iiii iiii iiii iiii
    0001 1001 0100 0000 0000 0000 0000 1001

    lw $13, 0($10)          # load values from matrices         | 0x0040 BC18
    1000 11ss ssst tttt iiii iiii iiii iiii
    1000 1101 0100 1101 0000 0000 0000 0000

    lw $14, 0($11)                                              | 0x0040 BC1C
    1000 11ss ssst tttt iiii iiii iiii iiii
    1000 1101 0110 1101 0000 0000 0000 0000

    mult $13, $14                                               | 0x0040 BC20
    0000 00ss ssst tttt 0000 0000 0001 1000
    0000 0001 1010 1110 0000 0000 0001 1000

    mflo $15                                                    | 0x0040 BC24
    0000 0000 0000 0000 dddd d000 0001 0010
    0000 0000 0000 0000 1111 1000 0001 0010

    add $12, $12, $15                                           | 0x0040 BC28
    0000 00ss ssst tttt dddd d000 0010 0000
    0000 0001 1001 1111 0110 0000 0010 0000

    addi $9, $9, -1         # decrement counter                 | 0x0040 BC2C
    0010 00ss ssst tttt iiii iiii iiii iiii
    0010 0001 0010 1001 1111 1111 1111 1111

    addiu $10, $10, 4       # adj mat1 ptr                      | 0x0040 BC30
    0010 01ss ssst tttt iiii iiii iiii iiii
    0010 0101 0100 1010 0000 0000 0000 0100

    addu $15, $15, $12      # adj mat2 ptr by offset            | 0x0040 BC34
    0000 00ss ssst tttt dddd d000 0010 0001
    0000 0011 1110 1100 1111 1000 0010 0001

    bgez $0, loop                                               | 0x0040 BC38
#   offset = distance - 1 => -9 - 1 = -10
    0000 01ss sss0 0001 iiii iiii iiii iiii
    0000 0100 0000 0001 1111 1111 1111 0110

loop_exit: sw $12, 20($19)      # load return value             | 0x0040 BC3C
                                # to system stack
    1010 11ss ssst tttt iiii iiii iiii iiii
    1010 1110 0110 1100 1111 1111 1110 1100

Answer:
    | address  |    instruction                        |
    ------------------------------------------------------
```

```
0x0040 BC0C: 0011 1100 0000 0001 1111 1111 1111 1111
0x0040 BC10: 0011 0100 0010 1100 1111 1111 1111 1111
0x0040 BC14: 0001 1001 0100 0000 0000 0000 0000 1001
0x0040 BC18: 1000 1101 0100 1101 0000 0000 0000 0000
0x0040 BC1C: 1000 1101 0110 1101 0000 0000 0000 0000
0x0040 BC20: 0000 0001 1010 1110 0000 0000 0001 1000
0x0040 BC24: 0000 0000 0000 0000 1111 1000 0001 0010
0x0040 BC28: 0000 0001 1001 1111 0110 0000 0010 0000
0x0040 BC2C: 0010 0001 0010 1001 1111 1111 1111 1111
0x0040 BC30: 0010 0101 0100 1010 0000 0000 0000 0100
0x0040 BC34: 0000 0011 1110 1100 1111 1000 0010 0001
0x0040 BC38: 0000 0100 0000 0001 1111 1111 1111 0110
0x0040 BC3C: 1010 1110 0110 1100 1111 1111 1110 1100

0x1000 41AC: 1111 1111 1111 1111 1111 1111 1100 0111
```

3. Decode the following machine code to MIPS Assembly Language Instructions.
   The address and hexadecimal code are shown below:

address	instruction	code
0x0040 0000:	0001 1001 1010 0000 0000 0000 0000 0011 0001 10ss sss0 0000 iiii iiii iiii iiii	blez $t5, 0x0040 0010
0x0040 0004:	0000 0001 0000 0010 0100 0000 0010 0000 0000 00ss ssst tttt dddd d000 0010 0000	add $t0, $$0, $v0
0x0040 0008:	0010 0001 0010 1010 0000 0000 0000 0001 0010 00ss ssst tttt iiii iiii iiii iiii	addi $t2, $t1, 1
0x0040 000c:	0000 0100 0000 0001 0000 0000 0000 0100 0000 01ss sss0 0001 iiii iiii iiii iiii	bgez $0, 0x0040 0020
0x0040 0010:	1010 1111 1010 1000 1111 1111 1111 0100 1010 11ss ssst tttt iiii iiii iiii iiii	sw $t0, -12($sp)
0x0040 0014:		
0x0040 0018:		
0x0040 001C:		
0x0040 0020:		

4. Convert the following statements to Machine Code given the related addresses.

address	code	instruction
0x0040 81AC:	jal r_in	0000 0000 0100 0000 1100 0110 1100 1100 <-- target 0000 11ii iiii iiii iiii iiii iiii iiii <-- instruction 0000 1100 0001 0000 0011 0001 1011 0011
0x	: ...	
0x0040 C6CC:	r_in: addu $9, $t9, $0	0000 00ss ssst tttt dddd d000 0010 0001 0000 0001 0010 0000 0100 1000 0010 0001