Michael Huang
CS 47 Section 01
Homework 01

1A. <mnemonic>, <rd>, <rs>, <rt> = add r12, r05, r10 # opcode: 0x2 / funct: 0x2
[opcode (4) | rs (4) | rt (4) | rd (4) | funct (4)] → [0x2 | 05 | 10 | 12 | 0x2]
[0x2 | 05 | 10 | 12 | 0x2] → [0010 | 0101 | 1010 | 1100 | 0010] → 100101101011000010_2
Convert 100101101011000010_2 to octal, group in 3-bit. [100 | 101 | 101 | 011 | 000 | 010]
[100 | 101 | 101 | 011 | 000 | 010] → 455302_8.

1B. <mnemonic>, <rd>, <rs>, <rt> = sub r04, r05, r15 # opcode: 0x2 / funct: 0x3
[opcode (4) | rs (4) | rt (4) | rd (4) | funct (4)] → [0x2 | 05 | 15 | 04 | 0x3]
[0x2 | 05 | 15 | 04 | 0x3] → [0010 | 0101 | 1111 | 0100 | 0011] → 00100101111101000011_2
00100101111101000011_2 to octal, group in 3-bit. [000 | 100 | 101 | 111 | 101 | 000 | 011]
[000 | 100 | 101 | 111 | 101 | 000 | 011] → 457503_8.

1C. <mnemonic> <rt>, <rs>, <immediate> = addi r10, r12, 0x3A # opcode: 0x4
[opcode (4bit) | rs (4bit) | rt (4bit) | immediate (8bit)] → [0x4 | 12 | 10 | 0x3A]
[0x4 | 12 | 10 | 0x3A] → [0100 | 1100 | 1010 | 00111010] → 01001100101000111010_2
01001100101000111010_2 to octal, group in 3-bit. [001 | 001 | 100 | 101 | 000 | 111 | 010]
[001 | 001 | 100 | 101 | 000 | 111 | 010] → 1145072_8.

1D. <mnemonic> <rt>, <rs>, <immediate> = ori r13, r03, 0x1B #opcode: 0x3
[opcode (4bit) | rs (4bit) | rt (4bit) | immediate (8bit)] → [0x3 | 03 | 13 | 0x1B]
[0x3 | 03 | 13 | 0x1B] → [0011 | 0011 | 1101 | 00011011] → 00110011110100011011_2
00110011110100011011_2 to octal, group in 3-bit. [000 | 110 | 011 | 110 | 100 | 011 | 011]
[000 | 110 | 011 | 110 | 100 | 011 | 011] → 636433_8.

1E. <mnemonic> <address> = jmp 0x23C # opcode: 0x5
[opcode (4bit) | address (16-bit)] → [0x5 | 0x23C] → [0101 | 0000001000111100]
[0101 | 0000001000111100] → 01010000001000111100_2 to octal, group in 3-bit.
01010000001000111100_2 → [001 | 010 | 000 | 001 | 000 | 111 | 100] → 1201074_8.

1F. <mnemonic> <address> = jal 0x100F # opcode: 0x6
[opcode (4bit) | address (16-bit)] → [0x6 | 0x100F] → [0110 | 0001000000001111]
[0110 | 0001000000001111] → 01100001000000001111_2 to octal, group in 3-bit.
01100001000000001111_2 → [001 | 100 | 001 | 000 | 000 | 001| 111] → 1410017_8.

2A. The muNote system has 7 symbols, therefore it is a base 7 system.

Symbols	Do	Re	Mi	Fa	So	La	Ti
Decimal weight	0	1	2	3	4	5	6
Sequence	Re (1)	Do (0)	La (5)	Ti (6)	La (5)	So (4)	
Index	5	4	3	2	1	0	
Calculation	$1 * 7^5$	$0 * 7^4$	$5 * 7^3$	$6 * 7^2$	$5 * 7^1$	$4 * 7^0$	Total: 18855_{10}

2B.

Number	Quotient	Remainder
987654321/7	141093474	3 (LSB)

Michael Huang
CS 47 Section 01
Homework 01

141093474/7	20156210	4
20156210/7	2879458	4
2879458/7	411351	1
411351/7	58764	3
58764/7	8394	6
8394/7	1199	1
1199/7	171	2
171/7	24	3
24/7	3	3
3/7	0	3 (MSB)

33321631443_7 = FaFaFaMiReTiFaReSoSoFa in muNote.

3A.
($a0-$a2) 3 * 4 bytes
($t0, $t1) are not saved across calls.
($s0-$s5) 6 * 4 bytes
($fp, $ra) 2 * 4 bytes
Additional 8 bytes for double word.
Minus 4 bytes for stack pointer.
(3 * 4) + (6 * 4) + (2 * 4) + 8 – 4 = 48 bytes.

3B.
```
subbi   $sp, $sp, 48
sw      $fp, 48($sp)
sw      $ra, 44($sp)
sw      $a0, 40($sp)
sw      $a1, 36($sp)
sw      $a2, 32($sp)
sw      $s0, 28($sp)
sw      $s1, 24($sp)
sw      $s2, 20($sp)
sw      $s3, 16($sp)
sw      $s4, 12($sp)
sw      $s5, 8($sp)
addi    $fp, $sp, 48
```

4.
<u>Original</u>:
```
.macro pop_and_add($argS, $arg1, $arg2)
addi $sp, $sp, -4
lw $arg1, 0($sp)
addi $sp, $sp, -4
lw $arg2, 0($sp)
add $argS, $arg1, $arg2
.end_macro
```

Michael Huang
CS 47 Section 01
Homework 01

```
.text
main:
        pop_and_add($s2, $s1,$s0)
        add $s0, $s1, $s2
        pop_and_add($s3, $s4,$s5)
        add $s5, $s4, $s3
        sub $s6, $s4, $s0
```

Expanded:

```
.text
main:
        addi $sp, $sp, -4
        lw $s1, 0($sp)
        addi $sp, $sp, -4
        lw $s0, 0($sp)
        add $s2, $s1, $s0
        add $0, $s1, $s2
        addi $sp, $sp, -4
        lw $s4, 0($sp)
        addi $sp, $sp, -4
        lw $s5, 0($sp)
        add $s3, $s4, $s5
        add $s5, $s4, $s3
        sub $s6, $s4, $s0
```

5A.

ld64bit $t0, 0x100A0015

0x100A001E	0xF1
0x100A001D	0xEF
0x100A001C	0xDE
0x100A001B	0xCD
0x100A001A	0x6A
0x100A0019	0xA5
0x100A0018	0x67
0x100A0017	0x6C
0x100A0016	0x65
0x100A0015	0x93
0x100A0014	0x61
0x100A0013	0x2A
0x100A0012	0xFA
0x100A0011	0x2A
0x100A0010	0x13
0x100A000F	0x2F

Michael Huang
CS 47 Section 01
Homework 01

Little Endian: 0xDECD6AA5676C6593

5B.
Big Endian: 0x93656C67A56ACDDE

6A.

Symbol	**Instruction Number**	**Calculation**	**Address**
main	000	0x00010000 + 000 * 4	0x00010000
main_L1	006	0x00010000 + 006 * 4	0x00010018
main_L2	008	0x00010000 + 008 * 4	0x00010020
V1	000	0x00100000	0x00100000
V2	001	0x00100000 + D (13) + 3	0x00100010

For .asciiz, add 1 byte to address for every character + 1 for termination, so “Hello world!” has 12 characters + 1 for termination for .asciiz. Since 13 is D in hex, V2 is at address 0x0010000D. However, 0x0010000D is not divisible by 4 as determined by .align 2 where it is alignment word and the address must be divisible by 4. So we must add until the address is divisible by 4. 0x00100000 + D = 1048576 + 13 = 1048589, which is not divisible by 4. 1048592 is divisible, which is 1048589 + 3. D is 13, add 3 so then 13 (D_{16}) + 3 = 16 (10_{16}), where 16 is divisible by 4. Since 0x00100000 + D + 3 = 0x00100010, it is now divisible by 4.

6B.

Instruction line number (number listed on the side of each line in the given problem)	**Instruction number relative to main, where main is 000**	**Line**
005	002	bne $t0, $t1, main_L1
008	005	j main_L2