課題１：アセンブリ言語プログラミングの基礎

s1300106 森響輝

課題1-1 S = A+B-C

```
          .data
A:        .word 31
B:        .word 53
C:        .word 11
S:        .word 0
          .text
main:     lw   $8, A # A, B, C をロード
          lw   $9, B
          lw   $10, C
          add  $11, $8, $9 # S = A+B
          sub  $11, $11, $10 # S = S-C
          sw   $11, S
exit: j   exit
```

- 実行結果 $8 = 0000001f $9 = 00000035 $10 = 0000000b $11 = 00000049

S($11) = 16*4 + 9 = 73 実際に S = 31 + 53 - 11 = 73 以上からプログラムが正常に動作されたことが確認できた。

課題1-2 ΣAi

```
         .data
N:       .word 10    # The length of Array
A:       .word 9     # A[0] = 9
         .word 3     # A[1] = 3
         .word 12
         .word 7
         .word 23
         .word 1
         .word 23
         .word 43
         .word 54    # A[8] = 54
         .word 31    # A[9] = 31
S:       .word 0
         .text

main:     or   $8, $0, $0       # i = 0
          lw   $9, N
          la   $10, A           # Aの何番目の要素を見るかアドレスで管理する
          or   $11, $0, $0      # s = 0

loop:     beq  $8, $9, loopend  # i == n なら loopend へ
```

```
        addi $8, $8, 1        # i++
        lw   $12, 0($10)      # $12 に $10  が参照する値を格納
        add  $11, $11, $12    # s += Ai
        addi $10, $10, 4      # 次の要素に参照をずらす
        j    loop

loopend: sw   $11, S          # sの値をストア

exit:    j    exit
```

- 実行結果 ループごとに$11の値が増加し、最終的に $11(S) = 000000ce になった。 ce = 16*12 + 14 = 206 実際 ΣAi = 206 であるからプログラムが正常に動作されたことが確認できた。

課題1-3

```
        .data
N:      .word 10    # The length of Array
A:      .word 9     # A[0] = 9
        .word 3     # A[1] = 3
        .word 12
        .word 7
        .word 23
        .word 1
        .word 23
        .word 43
        .word 54    # A[8] = 54
        .word 31    # A[9] = 31
B:      .space 40   # 配列B の格納先  大きさは40バイト
        .text

main:    or $8, $0, $0 # i = 0
         lw $9, N
         la $10, A # A, Bのアドレスをコピー
         la $11, B

loop:    beq  $8, $9, loopend
         addi $8, $8, 1
         lw   $12, 0($10) # Aの要素を取る
         sw   $12, 0($11) # Bにコピー
         addi $10, $10, 4 # A, B の参照をずらす
         addi $11, $11, 4
         j loop

loopend:

exit: j exit
```

- 実行結果 Aの各要素をBにコピーすることができた。 以下格納された値 ~ 0x00000009 0x00000003 0x0000000c 0x00000007 0x00000017 0x00000001 0x00000017 0x0000002b 0x00000036 0x0000001f ~ (Data Segments よりコピー)

課題1-4 バブルソート

課題のサンプルソースコードと同じ動作になるようにアセンブリを書いた

```
       .data
N:      .word 10    # The length of Array
A:      .word 9     # A[0] = 9
        .word 3     # A[1] = 3
        .word 12
        .word 7
        .word 23
        .word 1
        .word 23
        .word 43
        .word 54    # A[8] = 54
        .word 31    # A[9] = 31
        .text

main:    or $8, $0, $0      # i = 0
         lw $9, N           # $9 = n-1
         addi $9, $9, -1

         la $10, A

loop1:   beq $8, $9, exit
         lw $11, N          # $11 = j = n-2
         addi $11, $11, -2
         j loop2

loop2:   lw $12, $11        # $12に 現在のjの値を格納 (Aの何番目の値かを格納)
         add $12, $12, $12  # ４倍することで Aの先頭アドレスから何アドレス先の値が
a[j]か判断できるようになる
         add $12, $12, $12  # 4倍
         add $12, $12, $10  # Aの先頭アドレスが$10に格納されているので$12に足す
         lw $13, 0($12)     # $13 = a[j]
         lw $14, 4($12)     # $14 = a[j+1]
         slt $15, $14, $13  # A[j] > A[j+1] かどうか判定
         beq $15, 1 swap    # 上の行がtrueならswap
         beq $11, $8, loop2end # loopの終了
         addi $11, $11, -1 # j の更新
         j loop2

loop2end: addi $8, $8, 1
          j loop1

swap:    sw $14, 0($12) #swap
         sw $13, 4($12)
         beq $11, $8, loop2end
```

```
          addi $11, $11, -1
          j loop2

exit: j exit
```

- 実行結果 Aのデータが昇順にソートされた。 0x0000000a 0x00000001 0x00000003 0x00000007 0x00000009 0x0000000c 0x00000017 0x00000017 0x0000001f 0x0000002b 0x00000036