2020/05/10

コンピュータアーキテクチャ論_Ex01

s1260027　　Shunsuke Onuki

課題１：
3 つの変数と定数を含んだ式「S = (A ＋ B ー C) | 3」の計算を行うプログラムを作成し、xspim でシミュレーションを行う。

課題２：配列の総和計算
メモリ上に並んだ N 要素からなる配列の総和を求めるプログラムを作成し、xspim シミュレータで動作確認を行う。

課題３：配列のコピー
メモリ上に並んだN要素からなる配列 A の内容を別の配列 B にコピーするプログラムを作成し、シミュレータで動作確認を行う。

課題４：バブルソート
バブルソートのアルゴリズムを使って、配列 A に格納された N 個の整数を昇順にソートするプログラムをアセンブラで作成し、シミュレータで動作確認を行う。

課題１：
3 つの変数と定数を含んだ式「S = (A ＋ B ー C)｜3」の計算を行うプログラムを作成し、xspim でシミュレーションを行う。
（考え方）
各変数の初期値は以下のように設定する。

アドレス	データ(10 進数)
A	19
B	75
C	10
S	結果の格納

2 値の計算しかできないので、A+B, (A+B)-C, A+B-C, (A+B-C)｜3　の順に計算していく。

（プログラムとその説明）

```
    .data
A:  .word 19
B:  .word 75
C:  .word 10
S:  .word 0
    .text
main:
    lw  $8,A            #Aの値
    lw  $9,B            #Bの値
    lw  $10,C           #Cの値
    add $11,$8,$9       #$11 = $8 + $9
    sub $12,$11,$10     #$12 = $11 - $10
    ori $13,$12,3       #$13 = $12 | 3
    sw $13,S            #Sに$13を格納
exit:j exit
```

（結果）
57

課題２：配列の総和計算
メモリ上に並んだ N 要素からなる配列の総和を求めるプログラムを作成し、
xspim シミュレータで動作確認を行う。

（考え方）
Loop を使って配列の要素数繰り返す。
アドレスを次にずらしながら値を加算していく。

（プログラムとその説明）

```
        .data
N:      .word 10                        #配列の要素数
A:      .word 8                         #A[0] = 8
        .word 4                         #A[1] = 4
        .word 7
        .word 12
        .word 13
        .word 19
        .word 23
        .word 43
        .word 56                        #A[8] = 56
        .word 32                        #A[9] = 32
S:      .word 0
        .text
main:
        or  $8,$0,$0                    #i=0
        lw  $9,N                        #Nの値
        lw  $10,S                       #Sの値
        la  $11,A                       #配列Aのアドレス

        loop:
            beq  $8, $9, loopend        #$8==$9 ならば loopend
            lw   $12,0($11)             #A[$11]の値
            addi $8,$8,1                #i++
            add  $10,$10,$12            #$10 += $12
            addi $11,$11,4              #$11 += 4
            j loop
        loopend:

exit:j exit
```

（結果）
D9

課題３：配列のコピー
メモリ上に並んだN要素からなる配列 A の内容を別の配列 B にコピーするプログラムを作成し、シミュレータで動作確認を行う。

（考え方）
Loop 内で配列 A の各要素をロードして、その要素を配列 B に格納する。

（プログラムとその説明）

```
      .data
N:    .word 10    # The length of Array
A:    .word 8     # A[0] = 8
      .word 4     # A[1] = 4
      .word 7
      .word 12
      .word 13
      .word 19
      .word 23
      .word 43
      .word 56    # A[8] = 56
      .word 32    # A[9] = 32
B:    .space 40   # 配列B の格納先を確保する。大きさは40バイト(10ワード分)
      .text
main:
      lw  $8, N                     #Nの値
      la  $9, A                     #配列Aのアドレス
      la  $10, B                    #配列Bのアドレス
      or  $11, $0, $0               #i=0
      la  $13, 0($9)                #配列Aのアドレス
      la  $14, B                    #配列Bのアドレス

      loop:
            beq  $8, $11, loopend    #i == N ならば loopend
            lw   $12,0($13)          #A[0]の値
            sw   $12,0($10)          #A[0]の値をB[0]に格納
            addi $11,$11,1             #i++
            addi $13,$13,4           #配列Aのアドレスずらし
            addi $10,$10,4           #配列Bのアドレスずらし
            j loop
      loopend:


exit:j exit
```

（結果）

Data Segments

DATA				
[0x00005000]	0x0000000a	0x00000008	0x00000004	0x00000007
[0x00005010]	0x0000000c	0x0000000d	0x00000013	0x00000017
[0x00005020]	0x0000002b	0x00000038	0x00000020	0x00000008
[0x00005030]	0x00000004	0x00000007	0x0000000c	0x0000000d
[0x00005040]	0x00000013	0x00000017	0x0000002b	0x00000038
[0x00005050]	0x00000020	0x0000000a	0x00000008	0x00000004
[0x00005060]	0x00000007	0x0000000c	0x0000000d	0x00000013

課題４：バブルソート
バブルソートのアルゴリズムを使って、配列 A に格納された N 個の整数を昇順にソートするプログラムをアセンブラで作成し、シミュレータで動作確認を行う。

（考え方）
下記のアルゴリズムを参考に考える。

```
for(int i = 0; i < n-1; i++){
        for(int j = n-2; j >= i; j--){
            if(a[j] > a[j+1]) {
                tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
}
```

入れ替えたい２値はそれぞれレジスタにロードされているため、各レジスタからそれぞれ適切なアドレスに格納する

（プログラムとその説明）

```
        .data
N:      .word 10                        # The length of Array
A:      .word 8                         # A[0] = 8
        .word 4                         # A[1] = 4
        .word 7
        .word 12
        .word 13
        .word 19
        .word 23
        .word 43
        .word 56                        # A[8] = 56
        .word 32                        # A[9] = 32
B:      .space 40                       # 配列B の格納先を確保する。大きさは40バイト(10ワード分)
        .text
```

```
main:
      lw $t0, N                         #配列Aの要素数
      la $t1, A                         #配列Aのアドレス
      addi $t2, $t0, -1                 #N-1
      addi $t3, $t0, -2                 #N-2
      or $t4, $0, $0                    #i=0
      or $t5, $0, $t3                   #j=N-2

      loop1:
            slt $t6, $t4, $t2           #i<N-1 ならば $t6=0
            beq $t6, $0, exit           #$t6==0 ならば exit
            la $t1, A                   #配列Aのアドレスをロード
            addi $t1, $t1, 32           #配列Aの最後から2番目のアドレス
            or $t5, $0, $t2             #j=N-2

      loop2:
            slt $t6, $t4, $t5           #i<N-2 ならば $t6=0
            beq $t6, $0 loopend1        #$t6==0 ならば loopend1

            lw $t7, 0($t1)              #A[j]の要素
            lw $t8, 4($t1)              #A[j+1]の要素
            slt $t9, $t7, $t8           #A[j]<A[j+1] ならば 1 でなければ 0
            beq $t9, $0, loopend2       #$t9==0 ならば loopend2
            sw $t7, 4($t1)              #A[j+1]の要素
            sw $t8, 0($t1)              #A[j]の要素

      loopend2:
            addi $t1, $t1, -4           #A[j]の直前の要素のアドレス
            addi $t5, $t5, -1           #j--
            j loop2

      loopend1:
            addi $t4, $t4, 1            #i++
            j loop1

exit:
      j exit
```

(結果)

Data Segments

DATA				
[0x00005000]	0x0000000a	0x00000038	0x0000002b	0x00000020
[0x00005010]	0x00000017	0x00000013	0x0000000d	0x0000000c
[0x00005020]	0x00000008	0x00000007	0x00000004	0x00000000
[0x00005030]...[0x00005054]	0x00000000			
[0x00005054]	0x0000000a	0x00000008	0x00000004	
[0x00005060]	0x00000007	0x0000000c	0x0000000d	0x00000013
[0x00005070]	0x00000017	0x0000002b	0x00000038	0x00000020