2021/04/22

コンピュータアーキテクチャ論 Ex03

S1260027
Shunsuke Onuki

課題 3-1:行列の乗算

MIPS プログラミングとして行列の積を求めるプログラムを作ります。 メモリ上に格納されている２つの４ x ４行列の積を求めるプログラムを組み、 前回までと同じように xspim で動作確認をする。

課題 3-1:行列の乗算

MIPS プログラミングとして行列の積を求めるプログラムを作ります。 メモリ上に格納されている 2 つの 4 x 4 行列の積を求めるプログラムを組み、 前回までと同じように xspim で動作確認をする。 入力データには次の 2 つの行列を使用する。

0 0 0 1	0 1 2 3
0 2 0 0	4 5 6 7
0 0 3 0	8 9 10 11
4 0 0 0	12 13 14 15

(考え方)

C 言語で書かれた以下のプログラムを参考にする

```
#include <stdio.h>
main()
{
    static int mat1[4][4] = {
        { 0, 0, 0, 1 },
        { 0, 2, 0, 0 },
        { 0, 0, 3, 0 },
        { 4, 0, 0, 0 },
    };
    static int mat2[4][4] = {
        { 0, 1, 2, 3 },
        { 4, 5, 6, 7 },
        { 8, 9, 10, 11 },
        { 12, 13, 14, 15 },
    };
    static int result[4][4];
    int  i, j, k, s;
    /* 行列の乗算 */
    for( i = 0; i < 4; i++ ) {
        for( j = 0; j < 4; j++ ) {
            s = 0;
            for( k = 0; k < 4; k++ ) {
                s += mat1[i][k] * mat2[k][j];
            }
            result[i][j] = s;
        }
    }
    /* 結果の表示 */
    for( i = 0; i < 4; i++ ) {
        for( j = 0; j < 4; j++ ) {
            printf("%3d", result[i][j]);
        }
        printf("\n");
    }
}
```

（プログラムとその説明）

```
    .data
A: .word 0      #mat1[][]
   .word 0
   .word 0
   .word 1
   .word 0
   .word 2
   .word 0
   .word 0
   .word 0
   .word 0
   .word 3
   .word 0
   .word 4
   .word 0
   .word 0
   .word 0
B: .word 0      #mat2[][]
   .word 1
   .word 2
   .word 3
   .word 4
   .word 5
   .word 6
   .word 7
   .word 8
   .word 9
   .word 10
   .word 11
   .word 12
   .word 13
   .word 14
   .word 15
C: .space 64    #result[][]
N: .word 4

    .text

main:
    la $t0,A
    la $t1,B
    la $t2,C
    lw $t3,N                    # 4
    or $t4,$0,$0                # i
    or $t5,$0,$0                # j
    or $t6,$0,$0                # k
    or $t7,$0,$0                # s
```

```
jloop:
    beq $t3,$t5,jloopend        # j==4 なら jloopend
    or $t6,$0,$0                # k = 0
    or $t7,$0,$0                # s = 0

kloop:
    beq $t3,$t6,kloopend
    add $a0,$0,$t3              # a = 4
    add $a1,$0,$t4              # b = i
    jal MUL
    add $t8,$0,$v0              # $24 = i*4
    add $t8,$t8,$t6             # i*4 + k

    la $t0,A
    add $a0,$0,$t3              # a = 4
    add $a1,$0,$t8              # b = i*4+k
    jal MUL
    add $t0,$t0,$v0             # $8 = $8 + (i*4+k)*4
    lw $t8,0($t0)               # $24 = mat[i][k]

    add $a0,$0,$t3              # a = 4
    add $a1,$0,$t6              # b = k
    jal MUL
    add $t9,$0,$v0
    add $t9,$t9,$t5             #index of B (k*4 + j)

    la  $t1,B
    add $a0,$0,$t3              # 4
    add $a1,$0,$t9              # index of B
    jal MUL
    add $t1,$t1,$v0
    lw  $t9,0($t1)              #$25 = mat2[k][j]

    add $a0,$0,$t8              # a = mat1[i][k]
    add $a1,$0,$t9              # b = mat2[k][j]
    jal MUL
    add $t7,$t7,$v0             # s += mat1[i][k]*mat2[k][j]

    la $t0,A                    # Aのアドレス
    la $t1,B                    # Bのアドレス
    addi $t6,$t6,1              # k++
    j kloop


kloopend:
     add $a0,$0,$t3                 # a = 4
     add $a1,$0,$t4                 # b = i
     jal MUL
     add $t8,$0,$v0
     add $t8,$t8,$t5                # i*4 + j

     add $a0,$0,$t3                 # 4
     add $a1,$0,$t8                 # index of C
     jal MUL
     add $t8,$0,$v0                 #$24 = (i*4 + j) * 4
     la $t2,C
     add $t2,$t2,$t8
     sw $t7,0($t2)                  # result[i][j] = s
     add $t5,$t5,1                  # j++
     j jloop
```

```
jloopend:
     addi $t4,$t4,1                 #i++
     j iloop

iloopend:
     j exit

exit: j exit

MUL:
     or $s0,$0,$0
     addi $s0,$s0,1                 # mask
     addi $s6,$0,1
     or $s1,$0,$0                   # ans = 0

MUL_loop:
     slt $s7,$a1,$s0
     beq $s7,$s6,MUL_exit
     and $s5,$5,$s0                 #21 is tmp
     beq $s5,$0,MUL_loopend
     add $s1,$s1,$a0

MUL_loopend:
     addu $a0,$a0,$a0               #shift to left
     addu $s0,$s0,$s0               #shift to left
     j MUL_loop

MUL_exit:
     add $v0,$0,$s1
     jr $ra
```

（結果）