アセンブリ言語プログラミング(2)

第2週目

担当　情報システム部門　徳光政弘
2025年4月18日

今日の内容

- 2週目
  - 配列
  - サブルーチン
  - 簡単な関数の実現

配列

- data領域に配置する。
- 配列用に順番にデータが並ぶ。
- データのアクセスはアドレスになることに注意する。
- ラベルを使うとアドレスを参照できる。

ラベル

```
        .data

        .globl  v
v:      .word   1
        .word   2
        .word   3
        .word   4
```

1000	1
1004	2
1008	3
1012	4

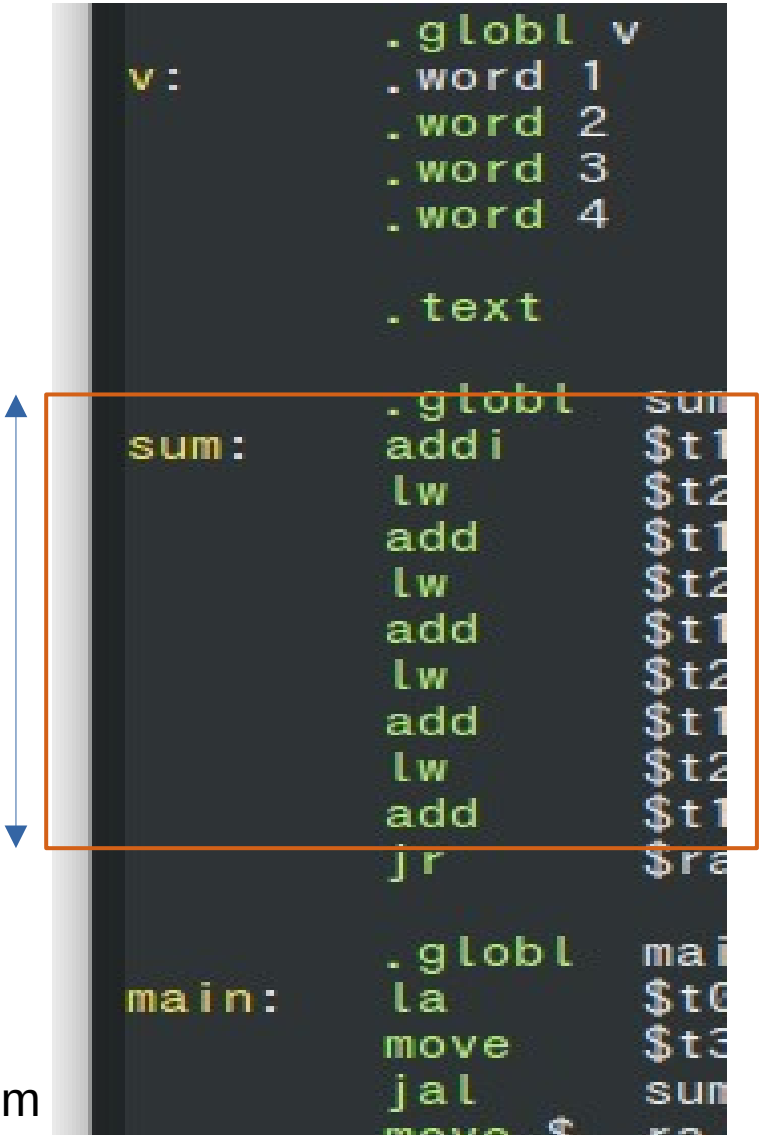
演習11

- メモリ上に配列に相当するデータを準備し、配列の総和を求める処理を実装する。総和の結果は$t9レジスタに保存する。
- 配列は右の数値とする。

v[0] = 3
v[1] = 2
v[2] = 1
v[3] = 4
v[4] = 5
v[5] = 8
v[6] = 17
v[7] = 12
v[8] = 10
v[9] = 5

サブルーチン

- 処理をひとまとめにしたものをサブルーチンと呼ぶ。
- 引数がない関数でもある。
- move命令で呼び出し側のアドレスをレジスタに待避する。
- jr命令で呼び出し側に戻ったときの命令を再開するアドレスをPCにセットする。



sample_14.asm

演習12

- メモリ上に配列に相当するデータを準備し、配列の総和を求めるサブルーチンを実装する。総和の結果は$t9レジスタに保存する。
- 配列は右の数値とする。

v[0] = 3
v[1] = 2
v[2] = 1
v[3] = 4
v[4] = 5
v[5] = 8
v[6] = 17
v[7] = 12
v[8] = 10
v[9] = 5

関数の実装方法

- レジスタの種類・個数には制限がある。
- 関数の呼び出し前、呼び出し後で勝手にレジスタの値は書き換えない工夫が必要となる。
- スタック領域にデータを待避して、関数内での処理が終わったら元に戻す。

関数の実装方法

```
        .data
        .text

        .globl  sum
sum:    addi    $sp, $sp, -4
        sw      $t0, 0($sp)
        addi    $t0, $zero, 0
        add     $t0, $a0, $t0
        add     $t0, $a1, $t0
        add     $t0, $a2, $t0
        move    $v0, $t0
        lw      $t0, 0($sp)
        addi    $sp, $sp, 4
        jr      $ra
```

```
        .globl main
main:   addi    $t0, $t0, 1
        addi    $t1, $t1, 2
        addi    $t2, $t2, 3
        addi    $t3, $zero, 10
        move    $a0, $t0
        move    $a1, $t1
        move    $a2, $t2
        move    $t5, $ra
        jal     sum
        move    $t4, $v0
        add     $t4, $t3, $t4
        move    $ra, $t5
        jr      $ra
```

```
#include <stdio.h>

int sum(int a, int b, int c) {
        return a + b + c;
}

int main(void) {
        int a = 1;
        int b = 2;
        int c = 3;
        int d = 10;

        int v = sum(a, b, c) + d;

        return 0;
}
```

sample_15.asm

$a0～$a3　引数用のレジスタ
$v0、$v1　戻り値用のレジスタ

関数の実装方法

- 詳細は教科書pp.100-104

演習13

- メモリ上に配列に相当するデータを準備し、配列の総和を求める関数sumを実装する。関数の引数は配列の先頭アドレスと個数とする。総和の結果は関数の呼び出し側で$t9レジスタに保存する。
- 配列の値は右のもとする。
- 引数
  - $a0　配列のアドレス
  - $a1　配列の大きさ

v[0] = 3
v[1] = 2
v[2] = 1
v[3] = 4
v[4] = 5
v[5] = 8
v[6] = 17
v[7] = 12
v[8] = 10
v[9] = 5