アセンブリ言語プログラミング(3)

第3週目

担当　情報システム部門　徳光政弘
2025年5月9日

今日の内容

- 3週目
  - 再帰を含む関数の実現

関数の再帰処理の考え方

- 教科書の補足　階乗計算の実装、スタックの使い方を補足

```c
int factorial(int n) {
    if (n == 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}

int main(void) {
    int n = 0;

    scanf("%d", &n);

    int v = factorial(n);
```

スタック領域の操作に関する部分

```
        .globl  fact
fact:   addi    $sp, $sp, -8
        sw      $ra, 0($sp)
        sw      $a0, 4($sp)
        slti    $t0, $a0, 1
        beq     $t0, $zero, L1
        addi    $v0, $zero, 1
        addi    $sp, $sp, 8
        jr      $ra

        .globl  L1
L1:     addi    $a0, $a0, -1
        jal     fact
        lw      $ra, 0($sp)
        lw      $a0, 4($sp)
        addi    $sp, $sp, 8
        mul     $v0, $a0, $v0
        jr      $ra
```

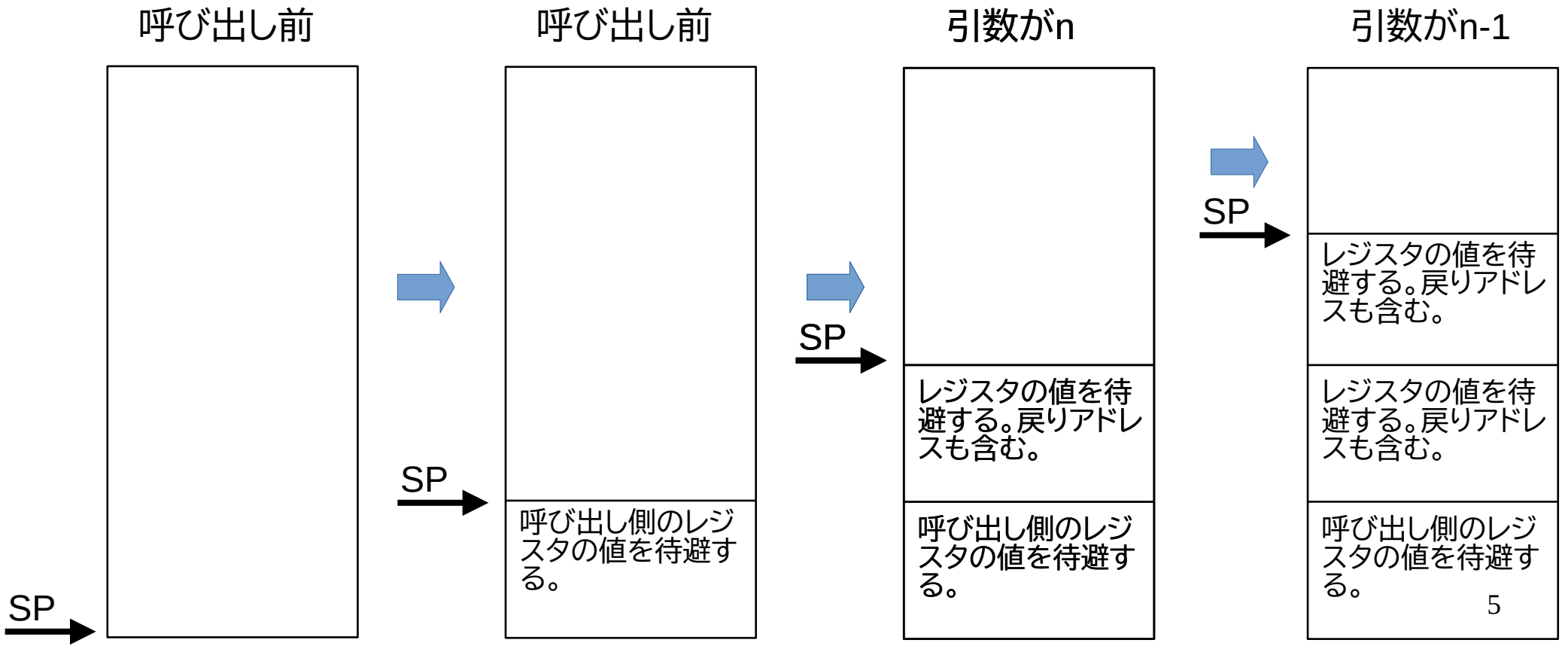
スタック領域の拡張

スタック領域の縮小

```
        .globl main
main:   addi    $t0, $t0, 4
        addi    $sp, $sp, -8
        sw      $ra, 0($sp)
        sw      $t0, 4($sp)
        move    $a0, $t0
        jal     fact
        move    $t1, $v0
        lw      $ra, 0($sp)
        lw      $t0, 4($sp)
        addi    $sp, $sp, 8
        jr      $ra
```

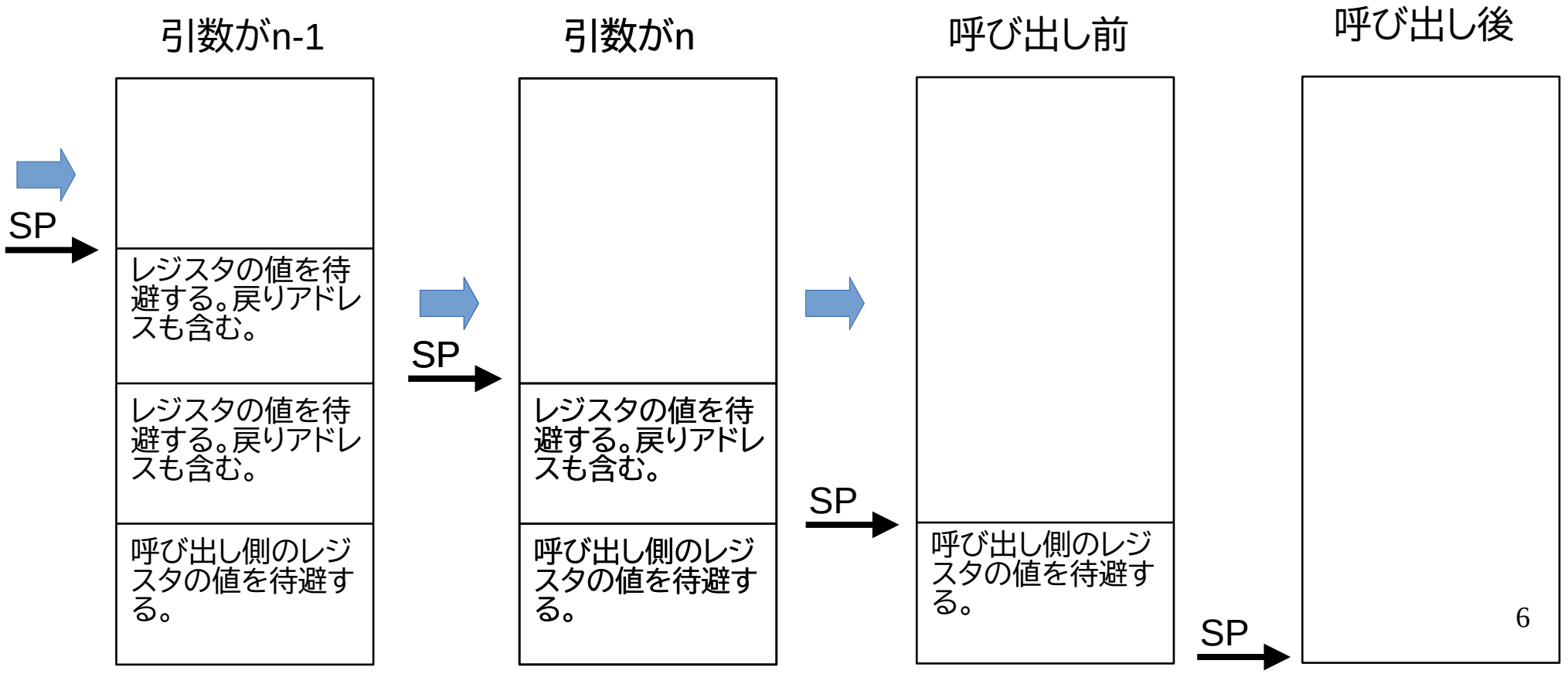
関数の再帰処理の考え方

スタック領域全体



関数の再帰処理の考え方

スタック領域全体



再帰を含む関数の実装方法

- 実装例はpp.105-106
- 例は階乗の計算
- 参考資料はWebClassにリンクあり

再帰を含む関数の実装方法

- 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。
- 再帰でさらに関数を呼び出すため、呼び出すときにはスタック領域にレジスタ上のデータを待避する。

演習14

- フィボナッチ数列の計算を再帰関数を使って実装する。フィボナッチ数列のC言語の処理は下記とする。フィボナッチ数列の引数は$t0レジスタ、演算結果は$t1レジスタに保存する。実行はn=10で動作確認する。

```
int fibonacci(int n) {
        int v = 0;

        if (n == 0) {
                v = 1;
        } else if (n == 1) {
                v = 1;
        } else {
                v = fibonacci(n - 1) + fibonacci(n - 2);
        }

        return v;
}
```

関数の実装方法

- 詳細は教科書pp.102-106

レポートの作成と提出

- すべての練習と演習のプログラムを作成する。
- レポートのひな型に沿って作成する。必要に応じて追記してよい。
- レポート本体のファイルは必ず提出すること。