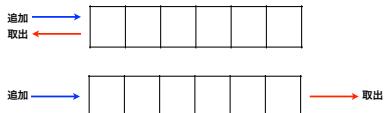


スタックと再帰関数

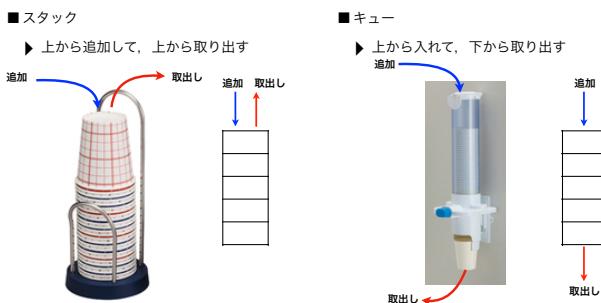
計算機構成の補足資料

1次元に並んだデータ構造に関する2つの操作

- 1次元に並んだデータ構造に対する追加と取り出しの操作について、2つの考え方がある。
- 片側だけを使い、追加と取出を行なう → 「スタック」
- 両側を使い、片方が追加、もう片方が取出を行なう。→ 「キュー」



実社会でのスタックとキューの例



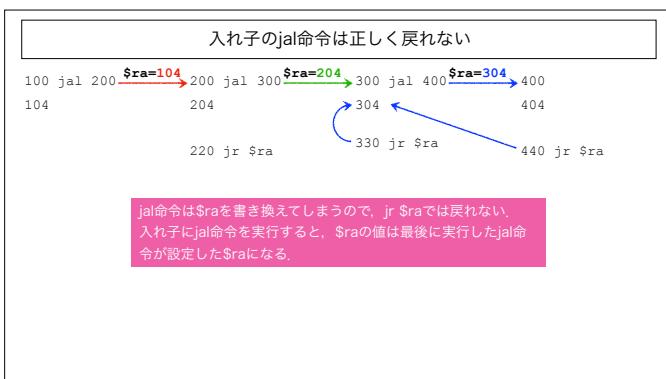
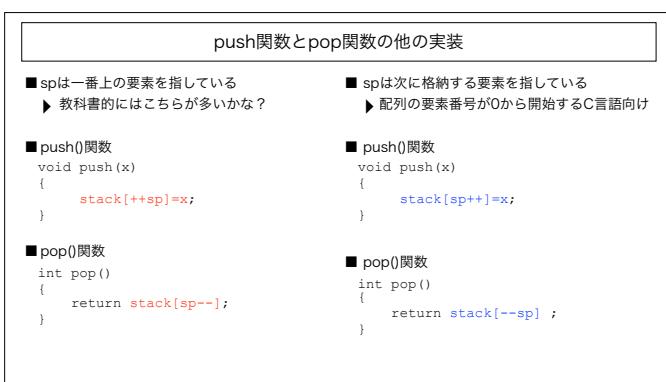
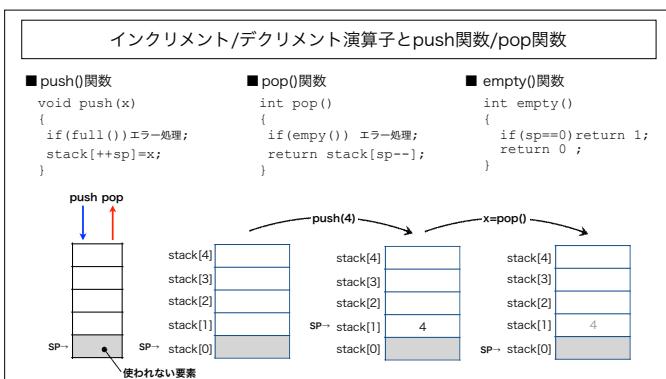
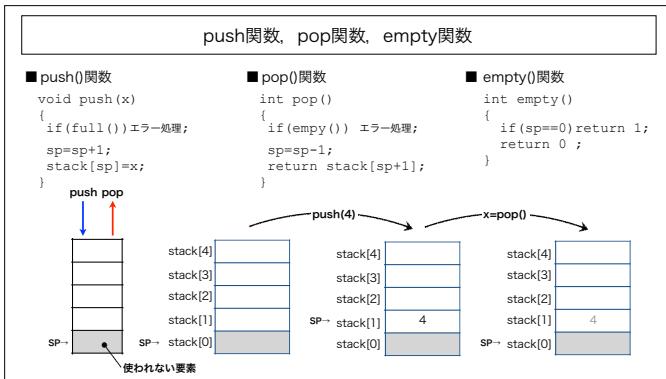
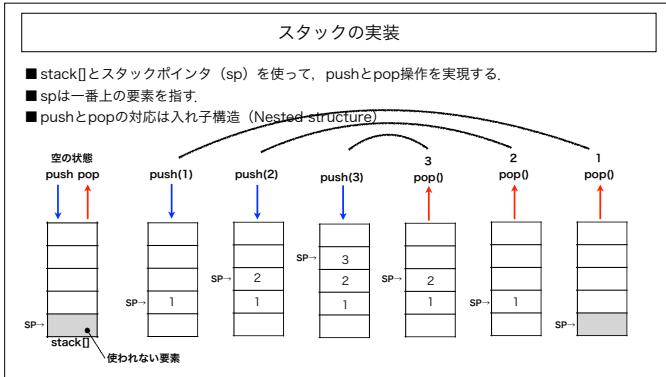
データ構造としてのスタックとキュー

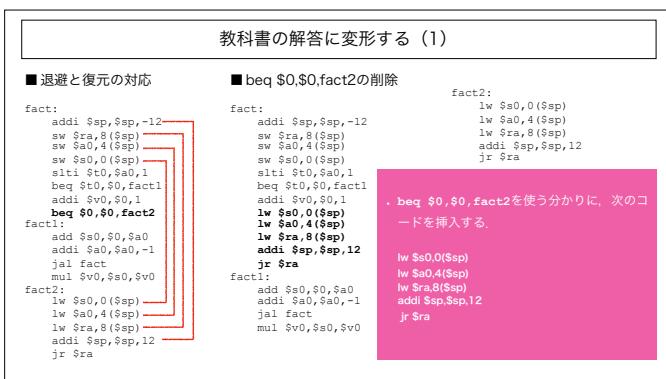
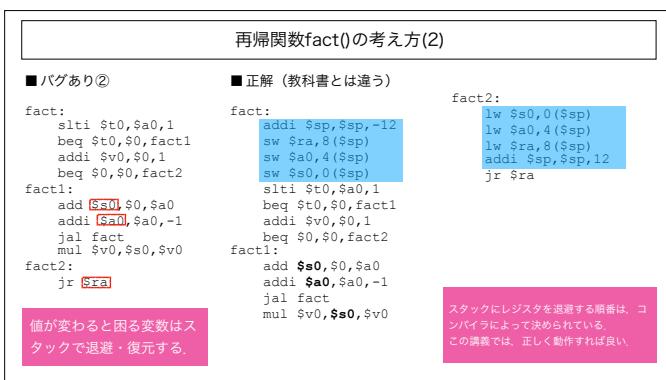
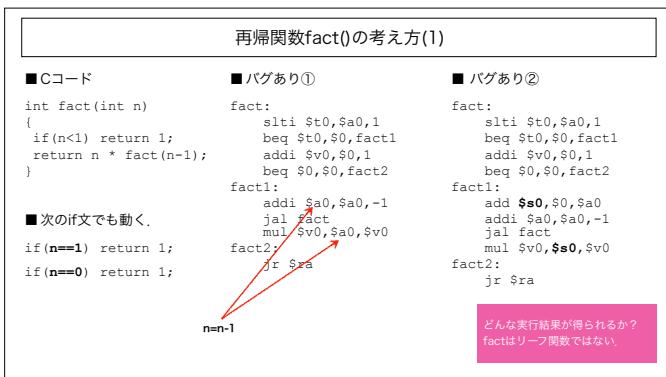
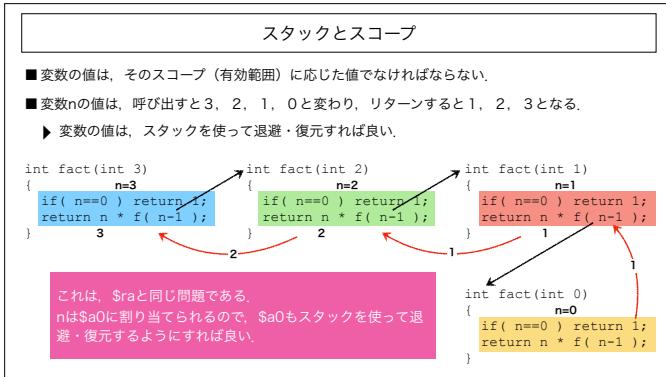
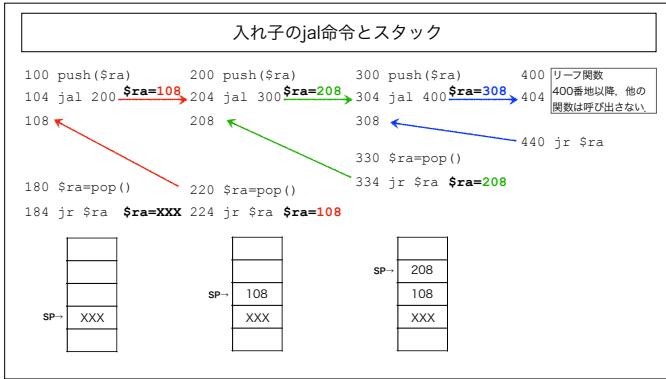
- | | |
|--|--|
| ■ スタック <ul style="list-style-type: none">▶ 片側だけを使って追加・取り出し▶ 上から追加して、上から取り出す▶ 新しいもの（最後に追加されたもの）が、最初に取り出される▶ LIFO Last In First Out<ul style="list-style-type: none">push追加取出しpop | ■ キュー <ul style="list-style-type: none">▶ 両側を使って追加・取り出し▶ 上から入れて、下から取り出す▶ 古いもの（最初に追加したもの）が、最初に取り出される▶ FIFO First In First Out<ul style="list-style-type: none">enqueue追加取出しdequeue |
|--|--|

スタックとキュー

スタックとキューが混在している







教科書の解答に変形する (2)

■ \$a0の値は復元されるので、復元後の

位置に次の命令を移動すると \$s0が不要になる。

- ▶ mul \$v0,\$s0,\$v0
- ▶ mul \$v0,\$a0,\$v0

■ mul命令の移動

```

fact1:
    addi $sp,$sp,-12
    sw $ra,8($sp)
    sw $a0,4($sp)
    sw $s0,0($sp)
    slli $t0,$a0,1
    beq $t0,$s0,1
    addi $v0,$0,1
    lw $s0,0($sp)
    lw $a0,4($sp)
    lw $ra,8($sp)
    addi $sp,$sp,12
    jr $ra

fact2:
    lw $s0,0($sp)
    lw $a0,4($sp)
    lw $ra,8($sp)
    addi $sp,$sp,12
    mul $v0,$a0,$v0
    jr $ra

fact1:
    addi $s0,$0,$a0
    addi $s0,$a0,-1
    jal fact
    mul $v0,$s0,$v0

```

教科書の解答に変形する (3)

■ \$s0は不要になるので、\$s0に関する

命令は削除できる。

- ▶ add \$s0,\$0,\$a0
- ▶ \$s0の退避・復元も削除

■ \$s0に関する命令の削除

```

fact1:
    addi $sp,$sp,-12
    sw $ra,8($sp)
    sw $a0,4($sp)
    sw $s0,0($sp)
    mul $v0,$a0,$v0
    jr $ra

fact2:
    -lw $s0,0($sp)
    addi $sp,$sp,-12
    lw $s0,4($sp)
    lw $ra,8($sp)
    addi $sp,$sp,12
    mul $v0,$a0,$v0
    jr $ra

fact1:
    add $s0,$0,$a0
    addi $s0,$a0,-1
    jal fact

```

教科書の解答に変形する (4)

■ \$s0に関する命令の削除に対応して、

スタックへの退避・復元のコードを修正する。

■ 退避・復元の修正

```

fact1:
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $a0,0($sp)
    sw $s0,0($sp)
    mul $v0,$a0,$v0
    slli $t0,$a0,1
    beq $t0,$s0,1
    addi $v0,$0,1
    addi $s0,$0,-1
    lw $s0,0($sp)
    lw $a0,4($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

fact2:
    lw $s0,0($sp)
    lw $a0,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    mul $v0,$a0,$v0
    jr $ra

fact1:
    addi $s0,$0,-1
    jal fact

```

教科書の解答に変形する (5)

■ 次のCコードに対応する部分で、\$a0

と \$raの値は変更されていないので復元は不要である。

- ▶ if(n<1) return 1;
- ▶ addi \$v0,\$0,1
- ▶ lw \$a0,0(\$sp)
- ▶ lw \$ra,4(\$sp)
- ▶ addi \$sp,\$sp,8
- ▶ jr \$ra

■ 無駄な命令の削除

```

fact1:
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $a0,0($sp)
    mul $v0,$a0,$v0
    slli $t0,$a0,1
    beq $t0,$s0,1
    addi $v0,$0,1
    addi $s0,$0,-1
    lw $s0,0($sp)
    lw $a0,4($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    jr $ra

fact2:
    lw $s0,0($sp)
    lw $a0,0($sp)
    mul $v0,$a0,$v0
    jr $ra

fact1:
    addi $s0,$0,-1
    jal fact

```

教科書の解答に変形する (6)

■ 見た目の違い

- ▶ L1とfact1
- ▶ fact2が残ってる
- mul \$v0,\$a0,\$v0は存在しない命令
- ▶ \$v0=\$a0×\$v0
- ▶ 32ビット×32ビットは64ビットになる
- 正しい命令列は次の通り。

```

mul $a0,$v0
mfhi $v0

```

■ 教科書の解答

```

fact:
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $a0,0($sp)
    slli $t0,$a0,1
    beq $t0,$s0,1
    addi $v0,$0,1
    addi $sp,$sp,8
    jr $ra

fact1:
    addi $s0,$0,-1
    jal fact
    addi $s0,$a0,-1
    jal fact
    fact2:
    lw $a0,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    mul $v0,$a0,$v0
    mul $v0,$a0,$v0
    jr $ra

```

■ 講義で示した解答

```

fact:
    addi $sp,$sp,-8
    sw $ra,4($sp)
    sw $a0,0($sp)
    slli $t0,$a0,1
    beq $t0,$s0,1
    addi $v0,$0,1
    addi $sp,$sp,8
    jr $ra

fact1:
    addi $s0,$0,-1
    jal fact
    fact2:
    lw $a0,0($sp)
    lw $ra,4($sp)
    addi $sp,$sp,8
    mul $v0,$a0,$v0
    mul $v0,$a0,$v0
    jr $ra

```