**Laporan Algoritma & Struktur Data**

**Pertemuan 9 : STACK**

Dosen Pengampu : Triana Fatmawati, S.T, M.T



Nama : Jami’atul Afifah

NIM : 2341760102

Prodi : D-IV Sistem Informasi Bisnis

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2023/2024**

1. Langkah-langkah pengerjaan dari beberapa rangkaian operasi stack:
2. `Push(10)`: Menambahkan 10 ke dalam stack. Stack saat ini: [10]
3. `Push(6)`: Menambahkan 6 ke dalam stack. Stack saat ini: [10, 6]
4. `Pop()`: Menghapus elemen teratas dari stack (6). Stack saat ini: [10]
5. `Push(8)`: Menambahkan 8 ke dalam stack. Stack saat ini: [10, 8]
6. `Push(2)`: Menambahkan 2 ke dalam stack. Stack saat ini: [10, 8, 2]
7. `Pop()`: Menghapus elemen teratas dari stack (2). Stack saat ini: [10, 8]
8. `Pop()`: Menghapus elemen teratas dari stack (8). Stack saat ini: [10]
9. `Push(4)`: Menambahkan 4 ke dalam stack. Stack saat ini: [10, 4]

Jadi, nilai top (elemen teratas) dari stack setelah langkah 8 adalah 4.

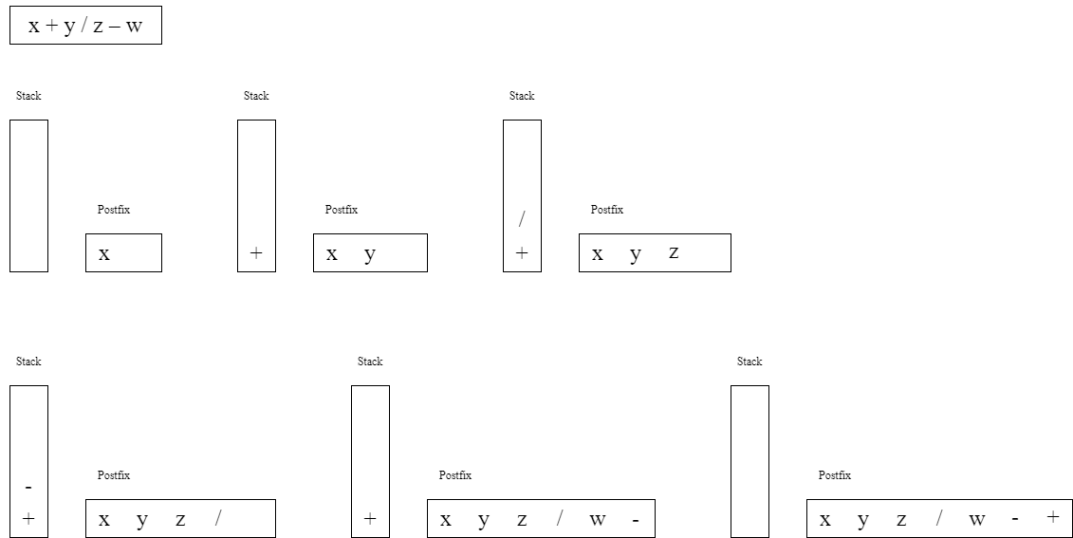
1. Lakukan konversi notasi infix berikut menjadi notasi postfix!
2. x + y / z – w = **x y z / w - +**

**Langkah - langkah V1 :**

1. Operand ‘x’, masuk ke postfix
2. Operan ‘+’, push ke stack dan operand ‘y’ masuk ke postfix.
3. Operand ‘/’, push ke stack dan operand ‘z’ masuk ke postfix.
4. Operand ‘w’, push ke posfix, dan operand ‘-’ push ke stack

Tetapi terlebih dahulu operand ‘/’ pop() dari stack dan masuk ke postfix (karena operand ‘/’ lebih diutamakan dari operand ‘-’).

1. Operand ‘-’ pop() dari stack dan masuk ke postfix.
2. Operand ‘+’ pop() dari stack dan masuk ke postfix.
3. Dan hasil akhirnya menjadi x y z / w - +.

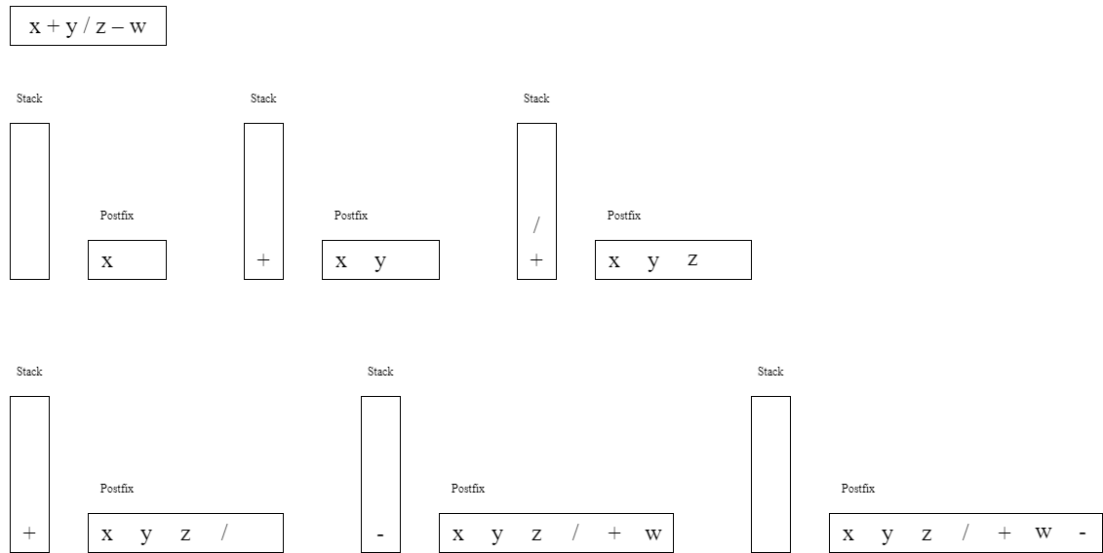


**Langkah - langkah V2:**

1. Operand ‘x’, masuk ke postfix
2. Operan ‘+’, push ke stack dan operand ‘y’ masuk ke postfix.
3. Operand ‘/’, push ke stack dan operand ‘z’ masuk ke postfix.
4. Operand ‘w’, push ke posfix, dan operand ‘-’ push ke stack

Tetapi terlebih dahulu operand ‘/’ pop() dari stack dan masuk ke postfix (karena operand ‘/’ lebih diutamakan dari operand ‘-’).

1. Operand ‘+’ pop() dari stack dan masuk ke postfix.
2. Operand ‘-’ pop() dari stack dan masuk ke postfix.
3. Dan hasil akhirnya menjadi x y z / w - +.



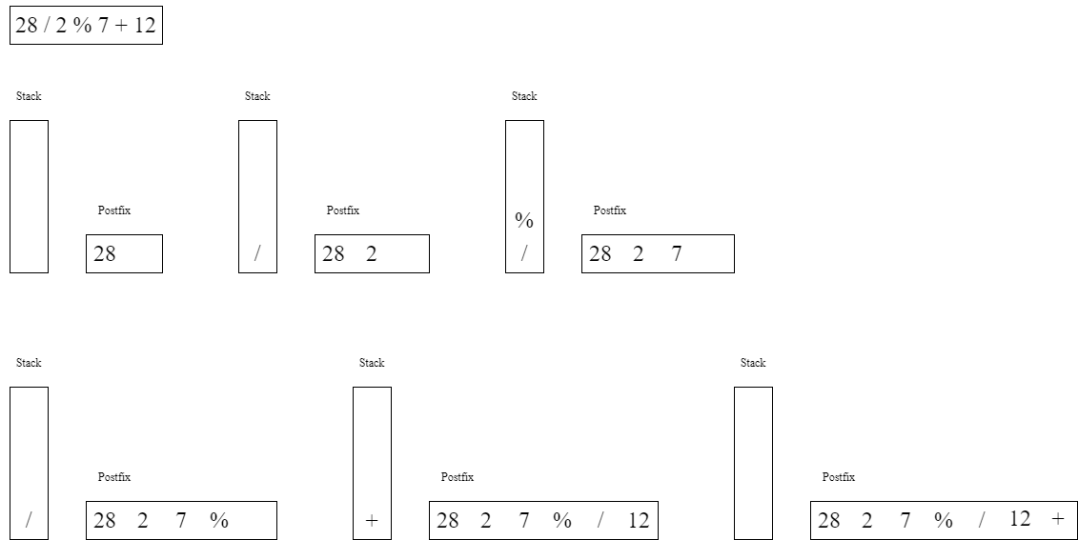
1. 28 / 2 % 7 + 12 = **28 2 7 % / 12 +**

**Langkah - langkah :**

1. Operand ‘28’, masuk ke postfix.
2. Operand ‘/’, push ke stack dan operand ‘2’, masuk ke postfix.
3. Operand ‘%’, push ke stack dan operand ‘7’, masuk ke postfix.
4. Operand ‘+’ push ke stack, dan operand ‘12’ masuk ke postfix

Tetapi terlebih dahulu operand ‘%’ dan oprand ‘/’, pop() dari stack dan masuk ke postfix (karena operand ‘%’ dan ‘/’ lebih diutamakan dari operand ‘+’).

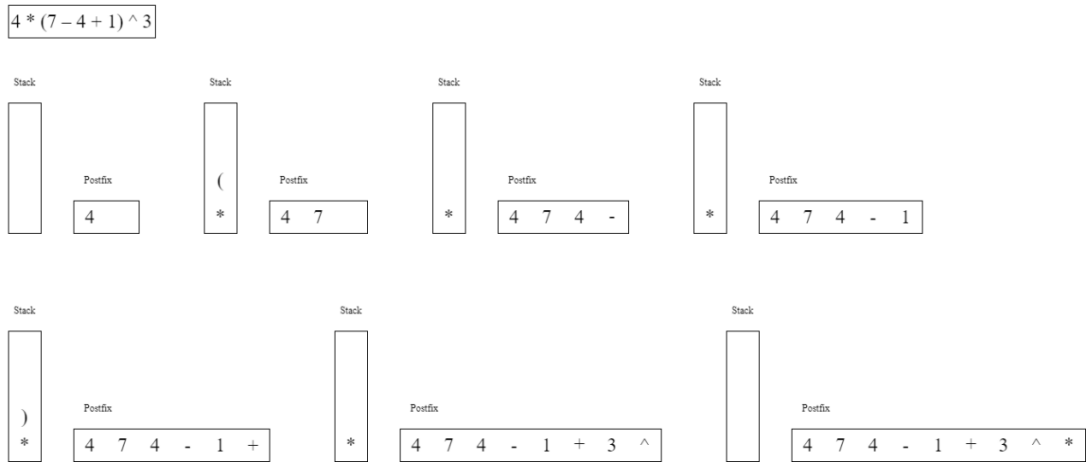
1. Operand ‘+’, pop() dari stack dan masuk ke postfix.
2. Dan hasil akhirnya menjadi 28 2 7 % / 12 +.



c. 4 \* (7 – 4 + 1) ^ 3 = **4 7 4 - 1 + 3 ^ \***

**Langkah - langkah :**

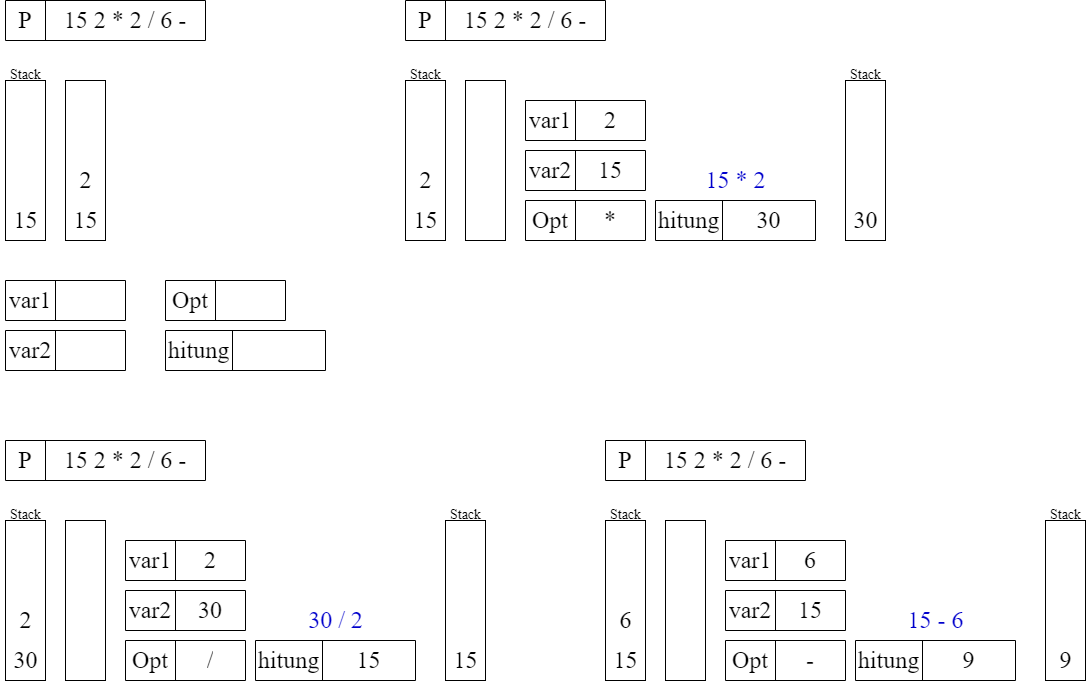
1. Operand ‘4’, masuk ke postfix.
2. Operand ‘\*’, push ke stack dan operand ‘7’, masuk ke postfix.
3. Operand ‘-’, push ke stack dan operand ‘4’, masuk ke postfix.
4. Operand ‘-’ pop() dari stack ke postfiks (karena operand ‘-’ berada dalam tanda kurung() sehingga lebih diutamakan).
5. Operand ‘+’, push ke stack dan operand ‘1’ masuk ke postfix.
6. Operand ‘+’, pop() dari stack ke postfix (karena operand ‘+’ berada dalam tanda kurung() sehingga lebih diutamakan).
7. Operand ‘^’, push ke stack dan operand ‘3’ masuk ke postfix.
8. Operand ‘^’, pop() dari stack ke postfix (karena operand pangkat ‘^’ lebih besar sehingga lebih diutamakan).
9. Operand ‘\*’, pop() dari stack ke postfix.
10. Dan hasil akhirnya menjadi 4 7 4 - 1 + 3 ^ \*.



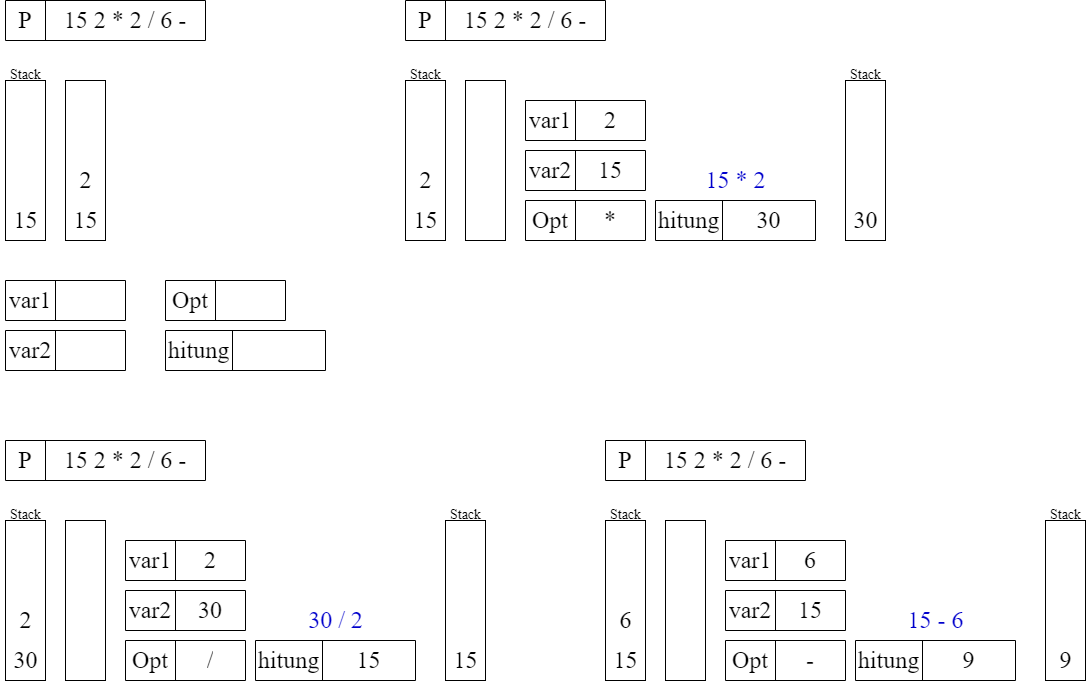
1. Hitung hasil ekspresi matematika berikut
2. 15 2 \* 2 / 6 –

**Langkah - langkah :**

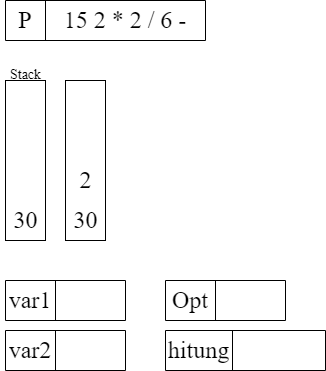
* 15 adalah operand, push ke stack.
* 2 adalah operand, push ke stack.



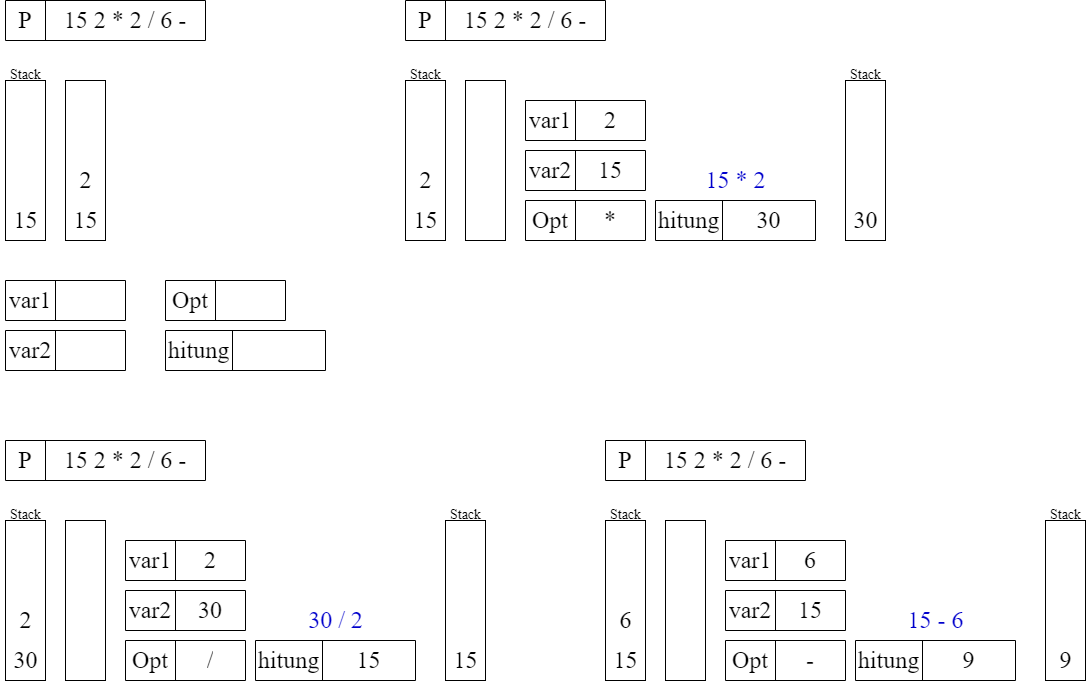
* \* adalah operator, pop 2 operand teratas dari stack (2 dan 15), simpan ‘2’ ke var1 dan simpan ‘15’ ke var2. Lakukan operasi **var2 opt var1** (15 \* 2 = 30), simpan hasilnya dalam variabel hitung, push variable hitung ke stack.



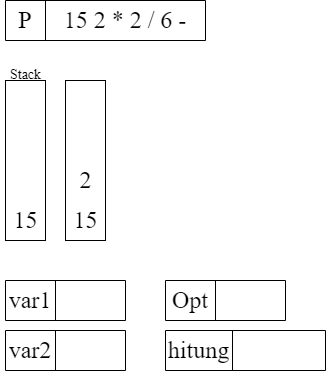
* 2 adalah operand, push ke stack



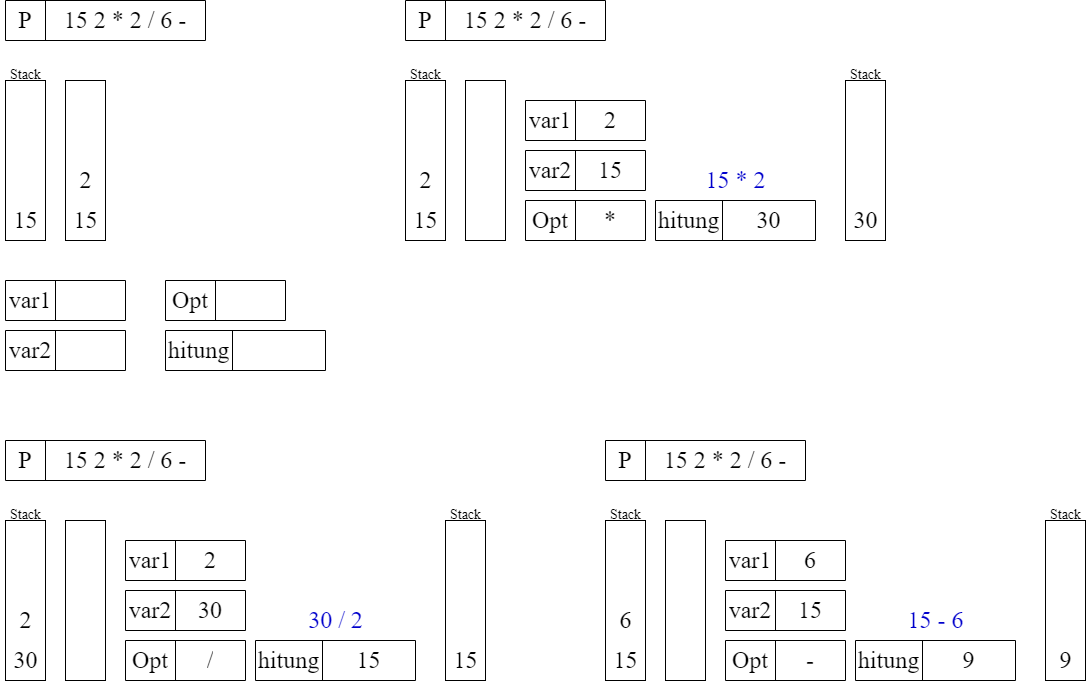
* / adalah operator, pop 2 operand teratas dari stack (2 dan 30), simpan ‘2’ ke var1 dan simpan ‘30’ ke var2. Lakukan operasi **var2 opt var1** (30 / 2 = 15), simpan hasilnya dalam variabel hitung, push variable hitung ke stack.



* 6 adalah operand, push ke stack



* - adalah operator, pop 2 operand teratas dari stack (6 dan 15), simpan ‘2’ ke var1 dan simpan ‘30’ ke var2. Lakukan operasi **var2 opt var1** (15 - 6 = 9), simpan hasilnya dalam variabel hitung, push variable hitung ke stack.

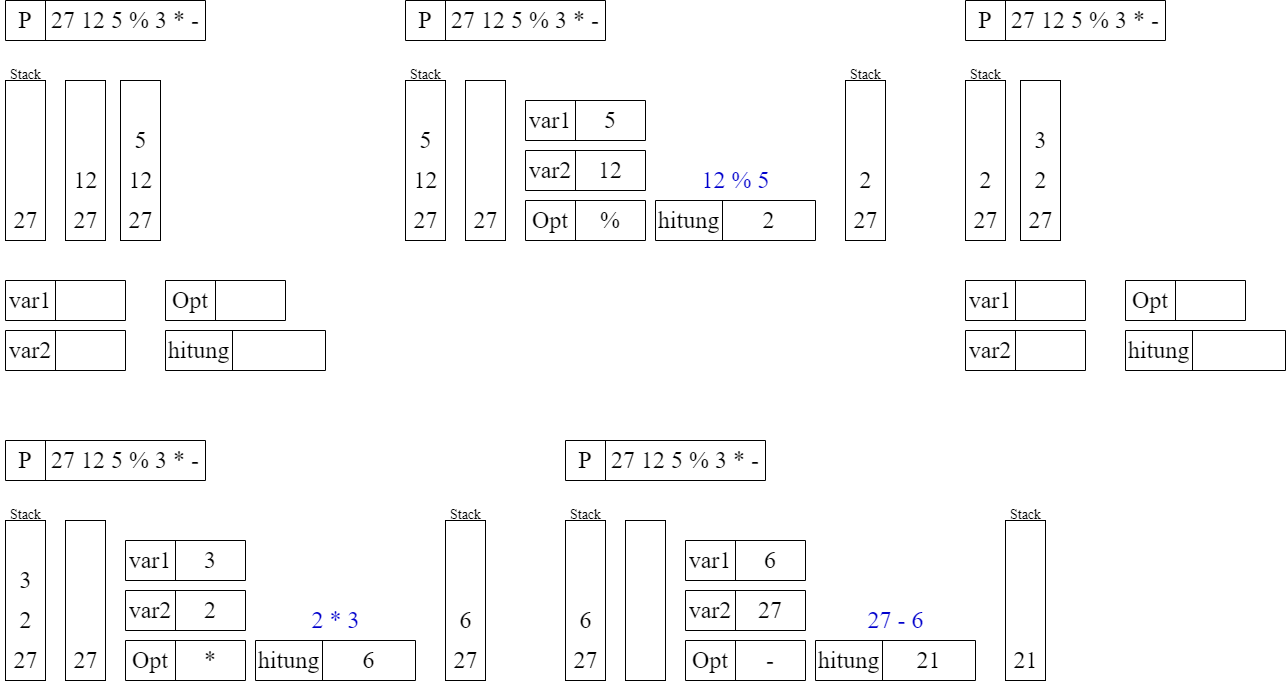


* Pop isi stack dan simpan di variable value sebagai hasil akhir.
* Persamaan 15 2 \* 2 / 6 –, hasil akhirnya adalah value = 9.

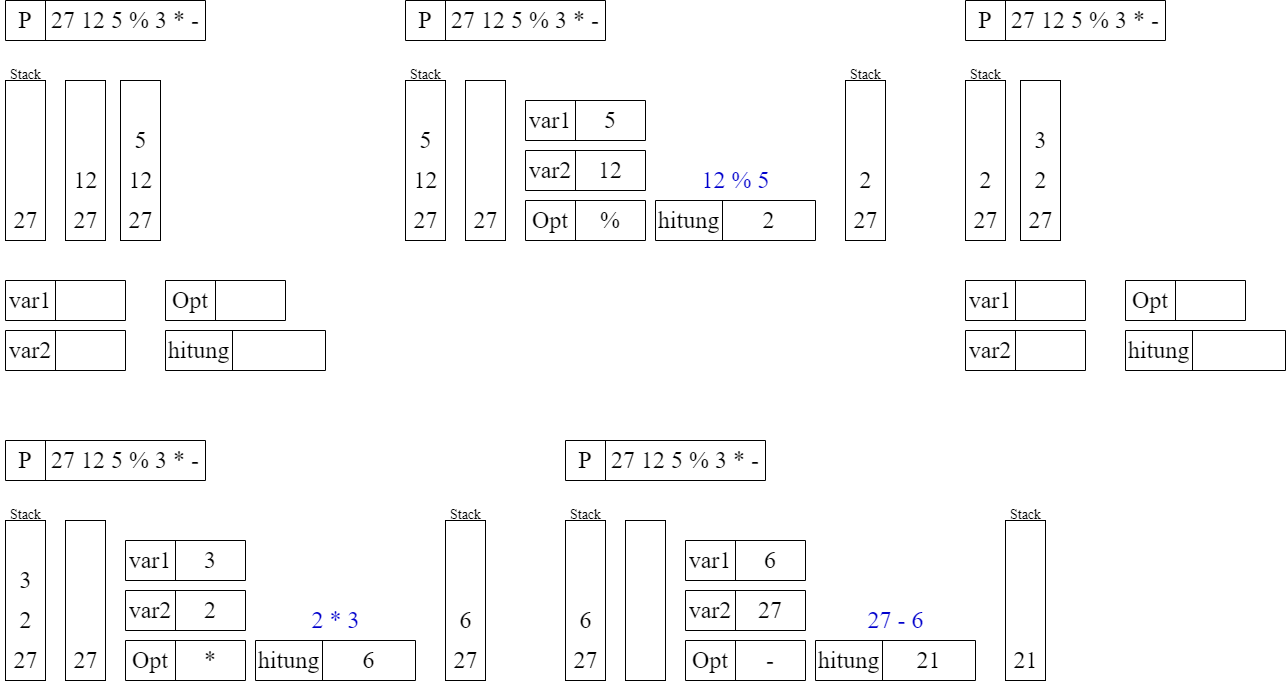
1. 27 12 5 % 3 \* –

**Langkah - langkah :**

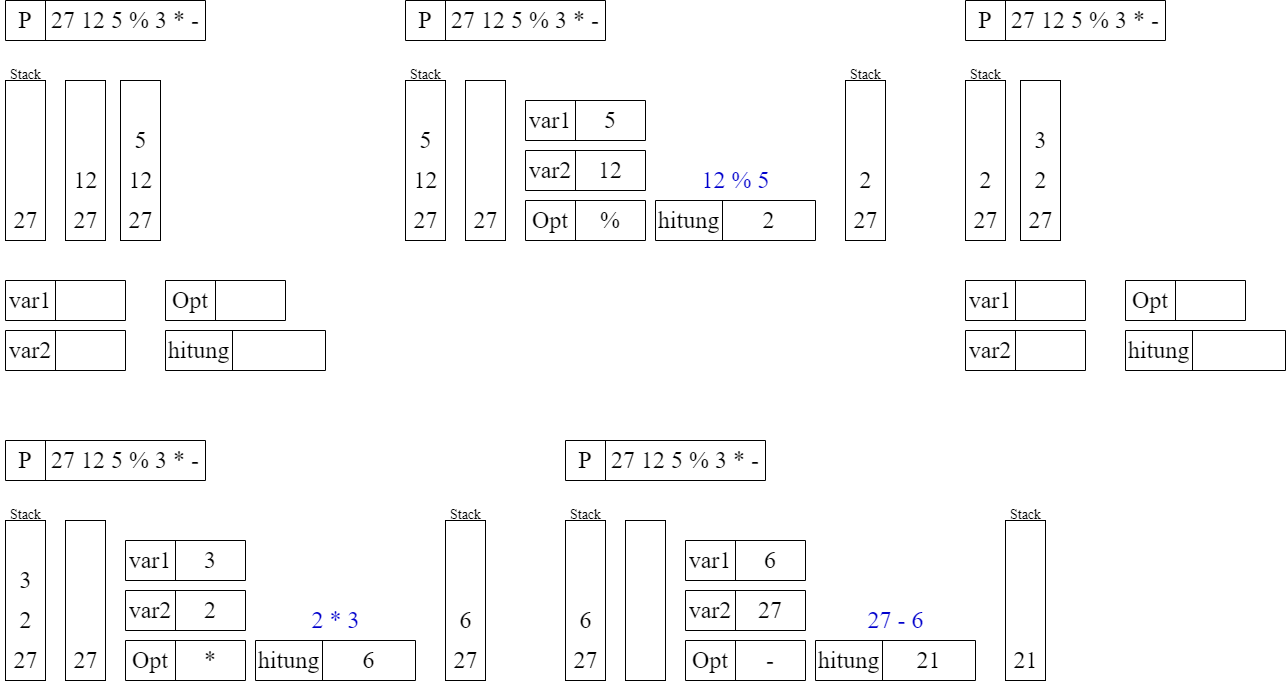
* 27 adalah operand, push ke stack.
* 12 adalah operand, push ke stack.
* 5 adalah operand, push ke stack.



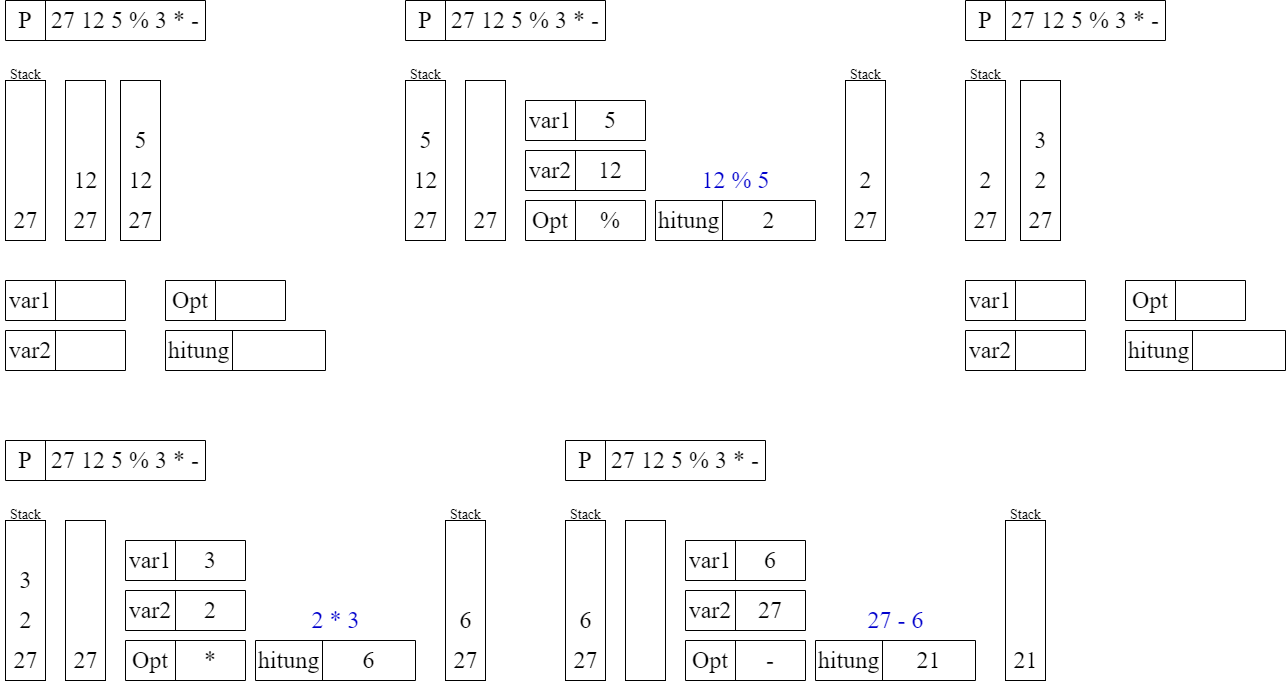
* % adalah operator, pop 2 operand teratas dari stack (5 dan 12), simpan ‘5’ ke var1 dan simpan ‘12’ ke var2. Lakukan operasi **var2 opt var1** (12 % 5 = 2), simpan hasilnya dalam variabel hitung, push variable hitung ke stack.



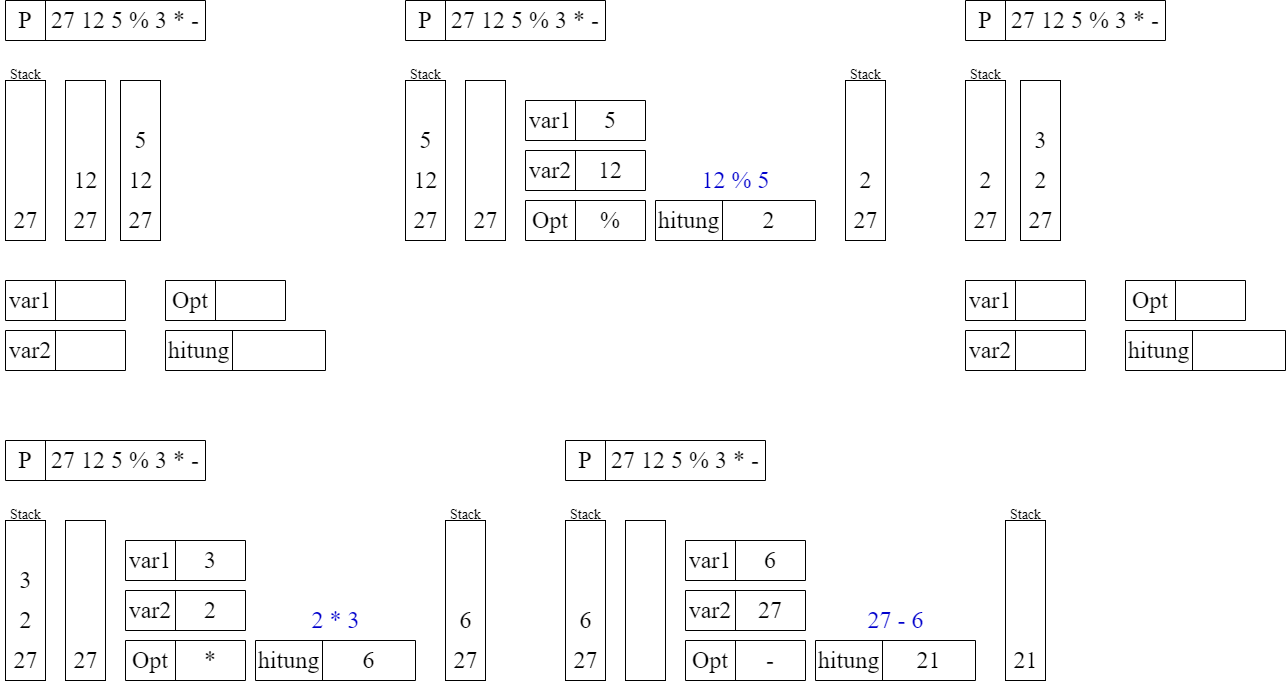
* 3 adalah operand, push ke stack.



* \* adalah operator, pop 2 operand teratas dari stack (3 dan 2), simpan ‘3’ ke var1 dan simpan ‘2’ ke var2. Lakukan operasi **var2 opt var1** (2 \* 3 = 6), simpan hasilnya dalam variabel hitung, push variable hitung ke stack.



* - adalah operator, pop 2 operand teratas dari stack (6 dan 27), simpan ‘6’ ke var1 dan simpan ‘27’ ke var2. Lakukan operasi **var2 opt var1** (27 - 6 = 21), simpan hasilnya dalam variabel hitung, push variable hitung ke stack.



* Pop isi stack dan simpan di variable value sebagai hasil akhir.
* Persamaan 27 12 5 % 3 \* –, hasil akhirnya adalah value = 21.