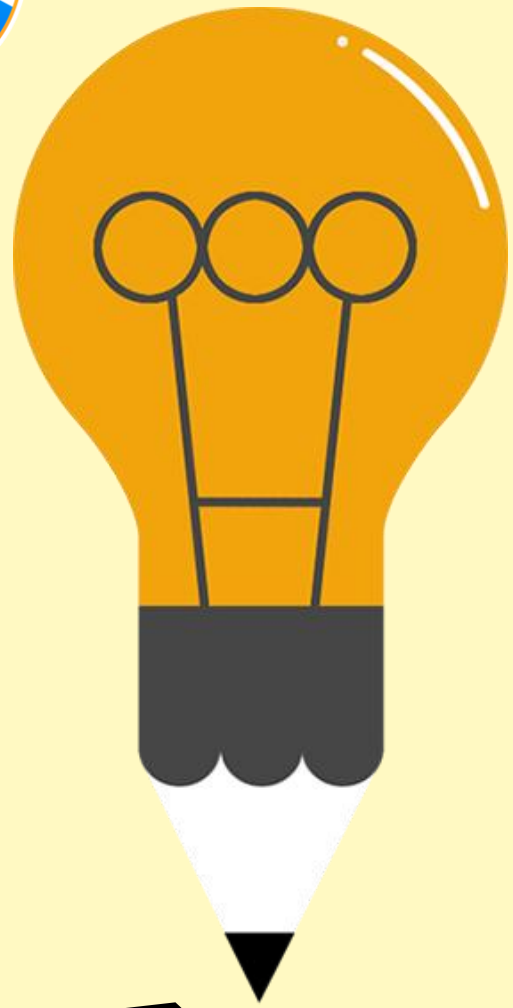


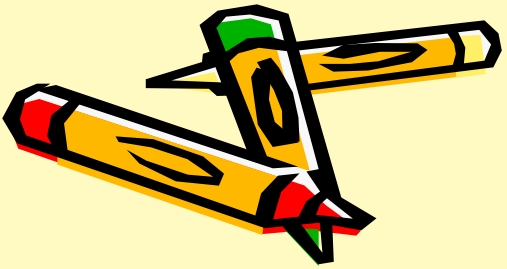
Code Optimization (Dasar)

Wakhid Kurniawan, S.Kom., M.Kom.

Program Studi S1 Informatika
Fakultas Sains dan Teknologi
Universitas Muhammadiyah Karanganyar

🎯 Tujuan Pembelajaran

- Setelah mempelajari topik ini, mahasiswa mampu:
- Menjelaskan konsep **Code Optimization** dan tujuannya dalam compiler.
- Memahami **teknik optimasi sederhana** pada kode menengah (TAC).
- Menerapkan contoh optimasi manual pada kode tiga alamat.

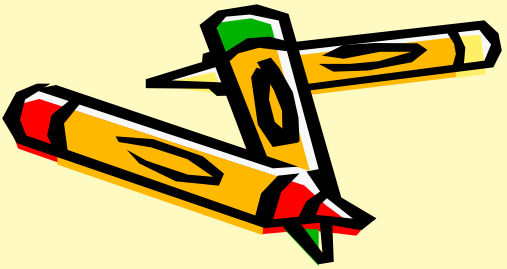


🧠 1. Apa itu Code Optimization?

"Code Optimization adalah proses **meningkatkan efisiensi** program tanpa mengubah hasil akhirnya."

Artinya, hasil program **tetap sama**, tapi:

- dijalankan **lebih cepat**,
- menggunakan **lebih sedikit memori**,
- dan **lebih efisien** saat dieksekusi.



⚙️ 2. Posisi Code Optimization dalam Alur Compiler

Source Code

↓

Lexical & Syntax Analysis

↓

Intermediate Code Generation

↓

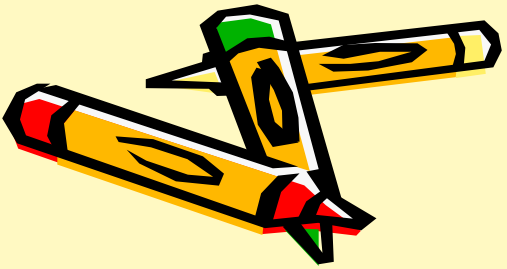
⚡ CODE OPTIMIZATION ⚡

↓

Target Code Generation (Assembly/Machine)

🧩 3. Tujuan Code Optimization

Tujuan	Penjelasan
⏱️ **Kecepatan eksekusi**	Menghapus operasi yang tidak perlu
💾 **Efisiensi memori**	Mengurangi variabel/temp yang berlebihan
🔁 **Reuse hasil perhitungan**	Hindari menghitung hal yang sama berulang kali
⚡ **Kinerja keseluruhan lebih baik**	Program jadi ringan dan cepat



🎬 4. Contoh Nyata (Tanpa Disadari Kita Sudah Melakukannya!)

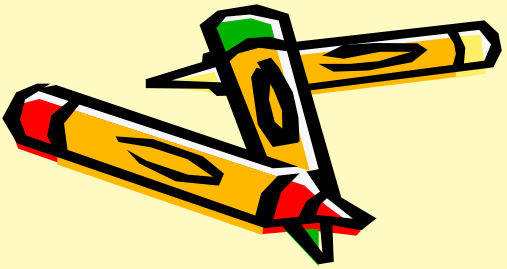
- Bayangkan kamu menulis kode Python:

```
for i in
range(len(data)):
    print(len(data))
```

Kompiler (atau otakmu 😄) tahu bahwa `len(data)` **tidak berubah**,
jadi kamu bisa menulis lebih efisien:

```
n = len(data)
for i in range(n):
    print(n)
```

Itu adalah **constant folding + loop optimization**! 🔥



💡 5. Jenis-Jenis Optimasi Sederhana

- Mari kita bahas **teknik dasar yang sering dipakai di tahap awal kompilasi.**
  - **1. Constant Folding (Penyatuan Konstanta)**
- **Sebelum:**

```
t1 = 3 * 4
t2 = t1 + x
```

- **Sesudah (Optimized):**

```
t2 = 12 + x
```

🧠 Compiler menyadari bahwa `3 * 4` adalah konstanta
→ langsung dihitung.

5. Jenis-Jenis Optimasi Sederhana…

- 2. Constant Propagation (Penyebaran Nilai Konstanta)

- **Sebelum:**

```
a = 5
b = a + 3
c = b * 2
```

- **Sesudah:**

```
b = 8
c = 16
```

Compiler mengganti semua `a` dengan nilainya (`5`).

💡 5. Jenis-Jenis Optimasi Sederhana…

- 3. Common Subexpression Elimination (CSE)

- **Sebelum:**

```
t1 = a + b
t2 = a + b
t3 = t1 * c
```

- **Sesudah:**

```
t1 = a + b
t3 = t1 * c
```

Compiler tahu bahwa `a + b` sudah dihitung sekali → tidak perlu ulangi lagi.

5. Jenis-Jenis Optimasi Sederhana…

4. Dead Code Elimination

- **Sebelum:**

```
a = 10
b = 20
c = a + b
a = 5   // a yang lama tidak pernah dipakai
```

- **Sesudah:**

```
b = 20
a = 5
```

Kompiler menghapus variabel atau pernyataan yang **tidak berpengaruh** ke hasil akhir.

5. Jenis-Jenis Optimasi Sederhana…

5. Strength Reduction

- Mengganti operasi "berat" dengan versi "lebih ringan".
- **Sebelum:**

```
t1 = i * 2
```

- **Sesudah:**

```
t1 = i + i
```

Karena penjumlahan lebih cepat dari perkalian
(terutama di hardware lama atau sistem embedded).

💡 5. Jenis-Jenis Optimasi Sederhana…

6. Loop Invariant Code Motion

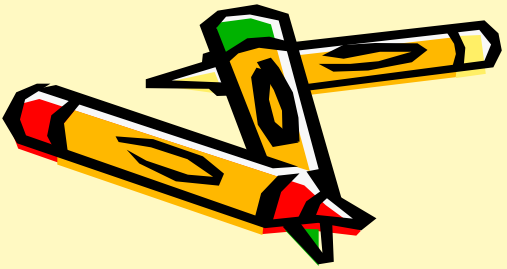
- **Sebelum:**

```
for i in range(0, n):
    t = a * b
    c[i] = t + i
```

- **Sesudah:**

```
t = a * b
for i in range(0, n):
    c[i] = t + i
```

Karena `a * b` **tidak berubah di setiap iterasi**, bisa dipindahkan ke luar loop.



5. Jenis-Jenis Optimasi Sederhana…

- 7. Contoh Lengkap dari TAC

- **Sebelum:**

```
t1 = a + b
t2 = a + b
t3 = t2 * c
t4 = 3 * 4
t5 = t3 + t4
```

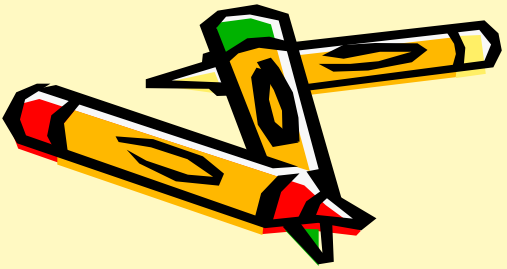
- **Sesudah:**

```
t1 = a + b          // CSE → hapus t2
t4 = 12             // Constant Folding
t5 = t1 * c + t4
```

🧮 7. Simulasi Interaktif

```
x = 2 * 3
y = 4 + x
```

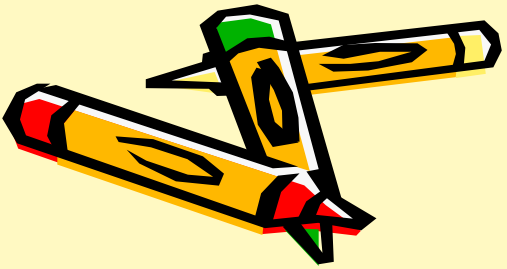
👉 Apa hasil setelah **constant folding**?



🧮 7. Simulasi Interaktif…

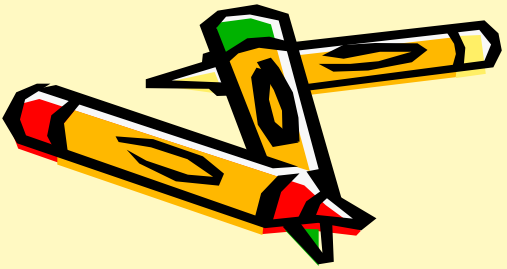
```
for i in range(5):
    x = a * b
    y = x + i
```

👉 Apa teknik optimasi yang relevan?



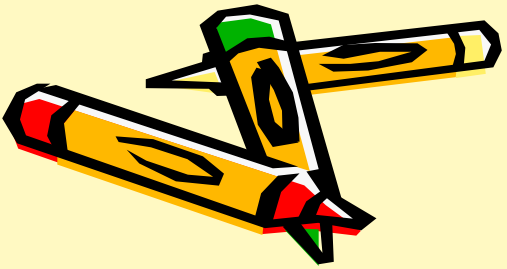
Ringkasan Materi

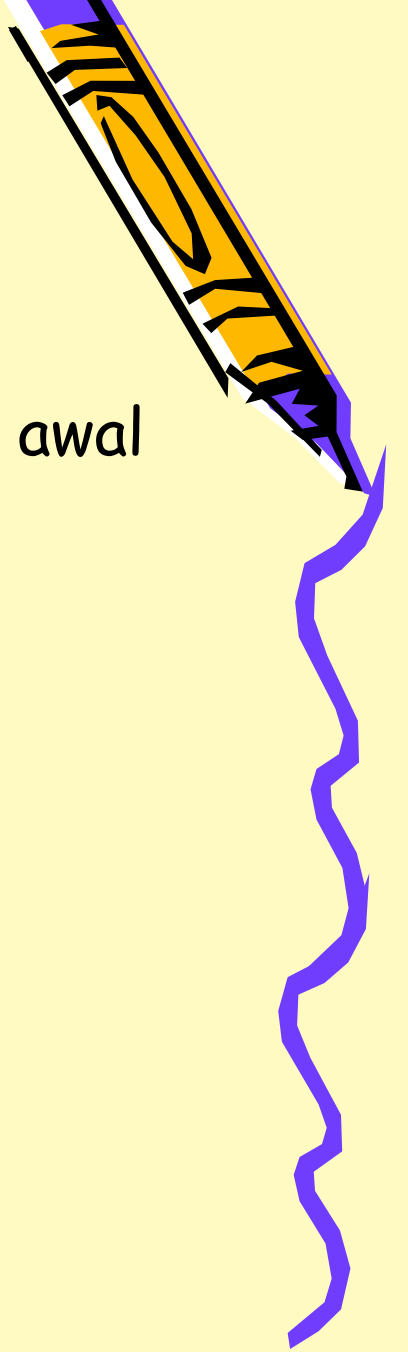
Konsep	Penjelasan
Code Optimization	Tahap membuat program lebih efisien tanpa ubah hasil
Tujuan utama	Kecepatan, efisiensi memori, dan kinerja
Teknik Dasar	Constant Folding, Propagation, CSE, Dead Code, Strength Reduction, Loop Optimization
Aturan Emas	Tidak boleh mengubah perilaku semantik program



🧮 Simulasi Interaktif

“Menurut kamu, lebih penting mana — menulis kode efisien sejak awal atau mengandalkan compiler untuk mengoptimasi?”





Tugas Latihan Mandiri

1. Lakukan optimasi pada kode tiga alamat berikut:

```
t1 = a + b
t2 = a + b
t3 = t1 * 2
t4 = 4 * 5
t5 = t3 + t4
```

- Jelaskan langkah optimasi yang dilakukan.

2. Buat contoh sederhana dari:
   - **Dead code elimination**
   - **Loop invariant code motion**