

# LAPORAN PRAKTIKUM

## ORGANISASI DAN ARSITEKTUR KOMPUTER

### PERCOBAAN 1

Nama : Akbar Dwi Herlambang  
NIM : 2308979  
Kelas : 2-C

- **Tugas Pendahuluan**

1. Setiap optimisasi memiliki kelebihan dan kekurangan nya masing-masing daripada optimisasi yang lain, misalkan diperbandingkan -O0 memiliki keunggulan dalam sedikitnya memory yang terpakai dan waktu yang digunakan untuk mengompilasi kode dibandingkan dengan optimisasi lain.

Sedangkan beda halnya dengan -O1 yang memiliki waktu eksekusi yang lebih cepat namun ukuran kode yang ditampung lebih sedikit, dan untuk -O2 merupakan optimisasi setingkat lebih tinggi dari -O1 yang memiliki waktu eksekusi setingkat lebih cepat, namun memiliki kelemahan dalam waktu kompilasi yang satu tingkat menjadi lebih tinggi dari -O1.

Peningkatan yang sama juga terjadi pada -O3 dimana optimisasi ini memiliki waktu kompilasi yang setingkat lebih lama daripada -O2 dan waktu eksekusi yang setingkat lebih cepat daripada -O2.

Optimisasi mulai terdapat perbedaan pada -Os, yang dimana optimisasi ini lebih bertujuan ke arah ukuran kode. Memiliki waktu kompilasi yang sama dengan -O2, dan ukuran kode yang lebih sedikit daripada optimisasi -O1 hingga -O3.

Optimisasi -Ofast memiliki kesamaan dengan -O3 namun terdapat peningkatan optimisasi terhadap fungsi-fungsi matematika yang tidak memerlukan akurasi tinggi. Memiliki tingkatan eksekusi dan waktu kompilasi yang sama dengan -O3.

2. Dari yang saya pahami, sepertinya hal ini dapat dilakukan karena perintah atau instruksi yang ditulis dari bahasa tingkat tinggi pada preprocessor ini di *compile* atau diterjemahkan kembali menjadi kode *assembly* yaitu kode basic yang universal dapat dipahami atau dieksekusi diberbagai platform.

3. a. EBP adalah register dari base pointer, dan berisikan current activation frame di stack. Sedangkan “pushl” ringkasnya berfungsi untuk mendorong, disini “pushl %ebp” memiliki fungsi untuk mendorong nilai EBP ke dalam stack.

Dan untuk “movl %esp %ebp” memiliki fungsi untuk memindahkan/movl nilai register ESP ke dalam register EBP.

b. Jikalau di dalam kode, digunakannya pertama kali prosedur *square* pada *squaresum* ketika `int temp1 = square(y);` jika diartikan/digambarkan dalam stack mungkin prosesnya adalah :

- “pushl %ebp” mendorong nilai EBP ke dalam stack
- lalu dilanjut ke baris kedua, dengan memindahkan nilai register ESP kedalam EBP
- baris ketiga yaitu “movl 8(%ebp), %eax” memindahkan nilai offset 8 byte dari alamat awal EBP pada stack dan menyimpan/menyalinnya kedalam EAX.

c. Prosedur Rekursif menurut saya memiliki kesamaan dengan fungsi loop yaitu perulangan, jadi sepemahaman saya ‘memory’ sendiri dalam konteks Prosedur Rekursif berfungsi sebagai tempat disimpannya variable atau nilai dalam proses rekursif/perulangan tersebut.

4. Dari yang saya pahami, stack atau bertumpuk, dan dipadukan dengan konteks dalam arsitektur komputer, memiliki arti atau maksud sebagai tumpukan atau activation fram yang disusun secara bertumpuk, dan setiap frame memiliki base pointer nya masing-masing yang didalam frame nya terdapat address dari tinggi ke rendah, dan base pointer akan menunjukkan frame dengan address yang tertinggi.

5. Dari apa yang saya pahami, hal ini disebut pemanggilan prosedur atau Procedure Call, dan apa yang terjadi pada *stack* pada saat pemanggilan prosedur diantara lain adalah memasukkan argument-argumen pemanggilan/callee dari awal hingga akhir secara berurutan ke dalam *stack*, memasukkan return address kedalam *stack*, memasukkan address dari old base pointer milik caller ke dalam stack. Memperbarui nilai base pointer yang sesuai dengan frame callee.

### • Tugas 1 : Proses Kompilasi Bahasa C Menggunakan GCC

#### 0. Source Code

```
// Modul      : 1
// Percobaan   : 1
// Tanggal    : 19 April 2024
// Nama (NIM)  : Akbar Dwi Herlambang (2308979)
// Nama File   : code.c
// Deskripsi   : Proses Kompilasi Bahasa C Menggunakan GCC

#define N_LOOP 500
int main(void)
{
    int indeks;
    int accumulator; indeks = 0;
    accumulator = 0; while(indeks<N_LOOP)
    {
        accumulator = accumulator + indeks;
        indeks = indeks + 1;
    }
    return accumulator;
}
```

1. Tampilan file code.c pada Notepad++

```

1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul : 1
3 // Percobaan : 1
4 // Tanggal : 19 April 2024
5 // Nama (NIM) : Akbar Dwi Herlambang (2308979)
6 // Nama File : code.c
7 // Deskripsi : Proses Kompilasi Bahasa C Menggunakan GCC
8
9 #define N_LOOP 500
10 int main(void)
11 {
12     int indeks;
13     int accumulator; indeks = 0;
14     accumulator = 0; while (indeks < N_LOOP)
15     {
16         accumulator = accumulator + indeks;
17         indeks = indeks + 1;
18     }
19     return accumulator;
20 }
21
22

```

2. Tampilan dari proses Preprocess – Compile – Assemble

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 1>gcc -E code.c > code.i
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 1>gcc -S code.c
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 1>gcc -c code.c
```

### 3. Tampilan code.exe pada HexEdit

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000:	5A	90	00	00	03	00	00	00	04	00	00	00	FF	FF	00	00
0010:	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
0020:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030:	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	00
0040:	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68
0050:	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	68	6F
0060:	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20
0070:	6D	64	46	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00
0080:	50	45	00	00	64	86	0F	00	E7	8F	22	66	00	68	00	00
0090:	88	04	00	00	F0	00	27	00	0B	02	02	1E	00	1E	00	00
00A0:	00	38	00	00	00	0A	00	00	E0	14	00	00	00	10	00	00
00B0:	00	00	40	00	00	00	00	00	00	10	00	00	00	02	00	00
00C0:	04	00	00	00	00	00	00	00	05	00	02	00	00	00	00	00
00D0:	00	20	01	00	00	04	00	00	CE	00	01	00	03	00	00	00
00E0:	00	00	20	00	00	00	00	00	00	10	00	00	00	00	00	00
00F0:	00	10	00	00	00	00	00	00	00	10	00	00	00	00	00	00
0100:	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00
0110:	00	80	00	00	50	07	00	00	00	00	00	00	00	00	00	00
0120:	00	50	00	00	70	02	00	00	00	00	00	00	00	00	00	00
0130:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0140:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0150:	40	40	00	00	28	00	00	00	00	00	00	00	00	00	00	00
0160:	00	00	00	00	00	00	00	00	CC	81	00	00	90	61	00	00
0170:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0180:	00	00	00	00	00	00	00	00	2E	74	65	78	74	00	00	00
0190:	D8	1C	00	00	00	10	00	00	00	1E	00	00	00	04	00	00
01A0:	00	00	00	00	00	00	00	00	00	00	00	00	60	00	50	60
01B0:	2E	64	61													

4. Tampilan code.o dan code.exe pada Notepad++

➤ code.o

[illegible]






➤ code.exe

```

1  M...
2
3  S...
4  N...
5  t...
6  b...
7  N...
8  N...
9  }...
10 '...
11 N...
12 r...
13 /...
14 /...
15 <...
16 N...
17 <...
18 +...
19 ~...
20 k...
21 i...
22 -...
23 A...
24 t...
25 N...
26 N...
27 R...
28 e...
29 l...
30 N...
31 f...
32 w...
33 N...
34 +...
35

```

## 5. Tampilan seluruh file dalam folder Tugas 1

Name	Date modified	Type	Size
 code.c	19/04/2024 22:30	C source file	1 KB
 code.exe	19/04/2024 22:38	Application	53 KB
 code.i	19/04/2024 22:30	I File	1 KB
 code.o	19/04/2024 22:31	O File	1 KB
 codes.s	19/04/2024 22:30	S File	1 KB

- **Tugas 2 : Proses Kompilasi Bahasa C Menggunakan GCC dengan Bantuan Batch File**

### 1. Tampilan kode batch.bat

```

1  %~d0
2  cd "%dp0"
3  gcc -E code.c > code.i
4  gcc -S code.c
5  gcc -c code.c
6  gcc -o code.exe code.c
7  code.exe
8
9  pause
10

```

### 2. Tampilan eksekusi batch.bat pada Command Prompt

```
C:\Windows\system32\cmd.exe

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>C:

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>cd "dp0"
The system cannot find the path specified.

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>gcc -E code.c 1>code.i

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>gcc -S code.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>gcc -c code.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>gcc -o code.exe code.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>code.exe

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 2>pause
Press any key to continue . . .
```

### 3. Hasil dari eksekusi batch.bat

Name	Date modified	Type	Size
batch.bat	19/04/2024 22:48	Windows Batch File	1 KB
code.c	19/04/2024 22:30	C source file	1 KB
code.exe	19/04/2024 22:49	Application	53 KB
code.i	19/04/2024 22:49	I File	1 KB
code.o	19/04/2024 22:49	O File	1 KB
code.s	19/04/2024 22:49	S File	1 KB

- **Tugas 3 : Dissassembly File Objek**

#### 1. Tampilan proses eksekusi Dissassembly code.o dan code.exe pada Command Prompt

```
C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 3>objdump -d code.o > disassembly_code_o.asm

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 3>objdump -d code.exe > disassembly_code_exe.asm
```

#### 2. Perbandingan antara File Dissassembly code.o dan code.exe pada Notepad++

➤ disassembly\_code\_o

```

disassembly_code_o.asm x disassembly_code_exe.asm x
1
2 code.o: file format pe-x86-64
3
4
5 Disassembly of section .text:
6
7 0000000000000000 <main>:
8 0: 55 push %rbp
9 1: 48 89 e5 mov %rsp,%rbp
10 4: 48 83 ec 30 sub $0x30,%rsp
11 8: e8 00 00 00 00 callq d <main+0xd>
12 d: c7 45 fc 00 00 00 00 movl $0x0,-0x4(%rbp)
13 14: c7 45 f8 00 00 00 00 movl $0x0,-0x8(%rbp)
14 1b: eb 0a jmp 27 <main+0x27>
15 1d: 8b 45 fc mov -0x4(%rbp),%eax
16 20: 01 45 f8 add %eax,-0x8(%rbp)
17 23: 83 45 fc 01 addl $0x1,-0x4(%rbp)
18 27: 81 7d fc f3 01 00 00 cmpl $0x1f3,-0x4(%rbp)
19 2e: 7e ed jle 1d <main+0x1d>
20 30: 8b 45 f8 mov -0x8(%rbp),%eax
21 33: 48 83 c4 30 add $0x30,%rsp
22 37: 5d pop %rbp
23 38: c3 retq
24 39: 90 nop
25 3a: 90 nop
26 3b: 90 nop
27 3c: 90 nop
28 3d: 90 nop
29 3e: 90 nop
30 3f: 90 nop
31

```

## ➤ disassembly\_code\_exe

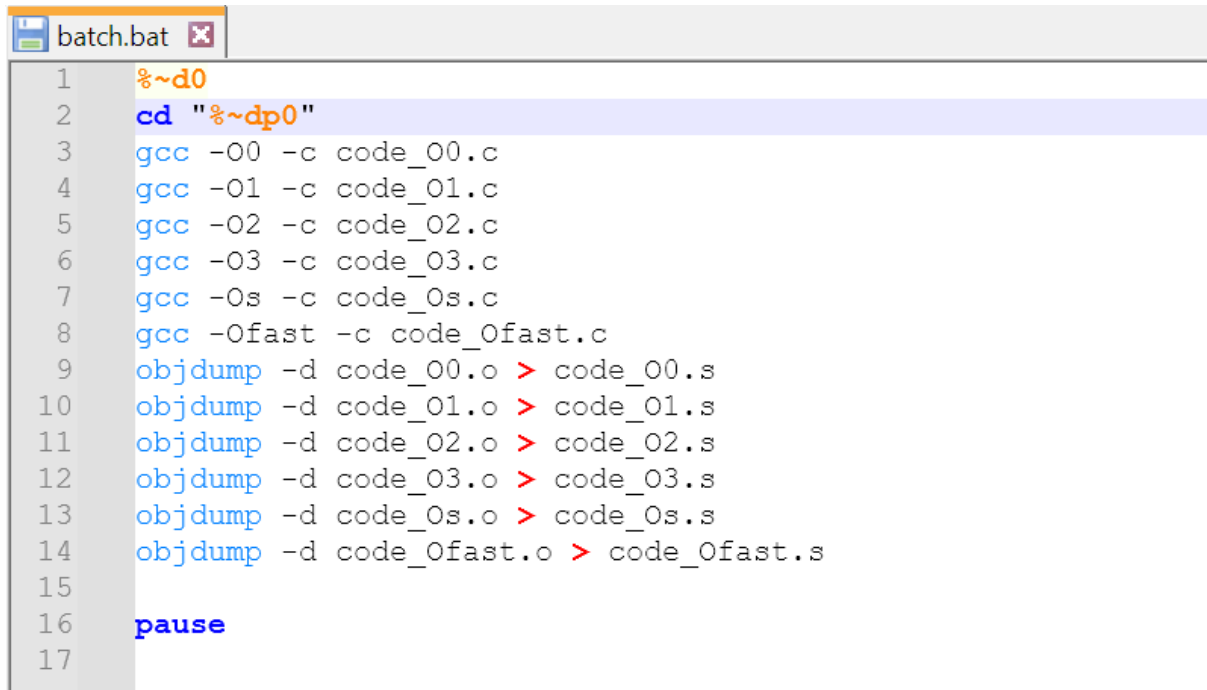
```

disassembly_code_o.asm x disassembly_code_exe.asm x
1
2 code.exe: file format pei-x86-64
3
4
5 Disassembly of section .text:
6
7 0000000000401000 <__mingw_invalidParameterHandler>:
8 401000: c3 retq
9 401001: 0f 1f 44 00 00 nopl 0x0(%rax,%rax,1)
10 401006: 66 2e 0f 1f 84 00 00 nopw %cs:0x0(%rax,%rax,1)
11 40100d: 00 00 00
12
13 0000000000401010 <pre_c_init>:
14 401010: 48 83 ec 28 sub $0x28,%rsp
15 401014: 48 8b 05 45 34 00 00 mov 0x3445(%rip),%rax # 404460 <.refptr.mingw_initlttsdrot_force>
16 40101b: 31 d2 xor %edx,%edx
17 40101d: c7 00 01 00 00 00 movl $0x1,(%rax)
18 401023: 48 8b 05 46 34 00 00 mov 0x3446(%rip),%rax # 404470 <.refptr.mingw_initlttsdyn_force>
19 40102a: c7 00 01 00 00 00 movl $0x1,(%rax)
20 401030: 48 8b 05 49 34 00 00 mov 0x3449(%rip),%rax # 404480 <.refptr.mingw_initlttsuo_force>
21 401037: c7 00 01 00 00 00 movl $0x1,(%rax)
22 40103d: 48 8b 05 0c 34 00 00 mov 0x340c(%rip),%rax # 404450 <.refptr.mingw_initcharmax>
23 401044: c7 00 01 00 00 00 movl $0x1,(%rax)
24 40104a: 48 8b 05 ef 32 00 00 mov 0x32ef(%rip),%rax # 404340 <.refptr.__image_base__>
25 401051: 66 81 38 4d 5a cmpw $0x5a4d,(%rax)
26 401056: 74 58 je 4010b0 <pre_c_init+0xa0>
27 401058: 48 8b 05 e1 33 00 00 mov 0x33e1(%rip),%rax # 404440 <.refptr.mingw_app_type>
28 40105f: 89 15 a3 5f 00 00 mov %edx,0x5fa3(%rip) # 407008 <managedapp>
29 401065: 8b 00 mov (%rax),%eax
30 401067: 85 c0 test %eax,%eax
31 401069: 74 35 je 4010a0 <pre_c_init+0x90>
32 40106b: b9 02 00 00 00 mov $0x2,%ecx
33 401070: e8 63 1a 00 00 callq 402ad8 <__set_app_type>
34 401075: e8 d6 1a 00 00 callq 402b50 <__p_fmode>
35 40107a: 48 8b 15 7f 33 00 00 mov 0x337f(%rip),%rdx # 404400 <.refptr._fmode>
36 401081: 8b 12 mov (%rdx),%edx

```

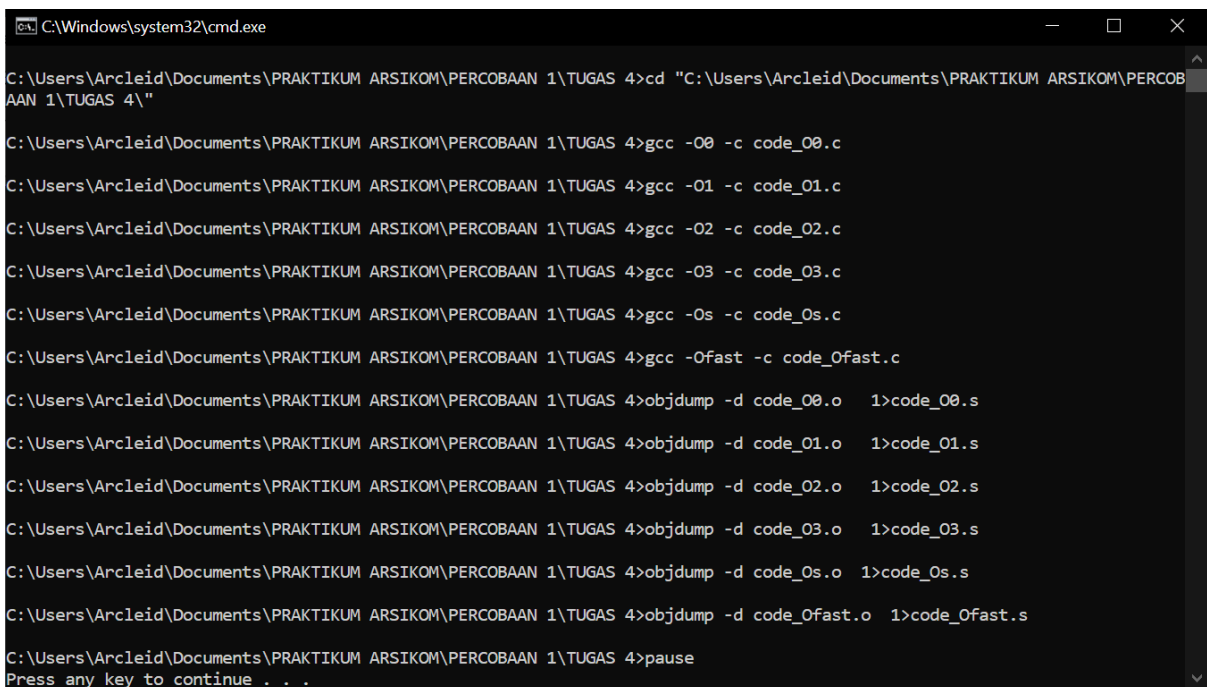
- **Tugas 4 : Optimisasi Kompilasi Program pada GCC**

1. Tampilan batch.bat



```
1 %~d0
2 cd "%~dp0"
3 gcc -O0 -c code_00.c
4 gcc -O1 -c code_01.c
5 gcc -O2 -c code_02.c
6 gcc -O3 -c code_03.c
7 gcc -Os -c code_0s.c
8 gcc -Ofast -c code_Ofast.c
9 objdump -d code_00.o > code_00.s
10 objdump -d code_01.o > code_01.s
11 objdump -d code_02.o > code_02.s
12 objdump -d code_03.o > code_03.s
13 objdump -d code_0s.o > code_0s.s
14 objdump -d code_Ofast.o > code_Ofast.s
15
16 pause
17
```

2. Tampilan eksekusi batch.bat pada Command Prompt



```
C:\Windows\system32\cmd.exe
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>cd "C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4\"
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>gcc -O0 -c code_00.c
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>gcc -O1 -c code_01.c
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>gcc -O2 -c code_02.c
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>gcc -O3 -c code_03.c
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>gcc -Os -c code_0s.c
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>gcc -Ofast -c code_Ofast.c
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>objdump -d code_00.o 1>code_00.s
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>objdump -d code_01.o 1>code_01.s
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>objdump -d code_02.o 1>code_02.s
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>objdump -d code_03.o 1>code_03.s
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>objdump -d code_0s.o 1>code_0s.s
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>objdump -d code_Ofast.o 1>code_Ofast.s
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 4>pause
Press any key to continue . . .
```

Name	Date modified	Type	Size
batch.bat	19/04/2024 23:01	Windows Batch File	1 KB
code_O0.c	19/04/2024 22:30	C source file	1 KB
code_O0.o	19/04/2024 23:01	O File	1 KB
code_O0.s	19/04/2024 23:01	S File	2 KB
code_O1.c	19/04/2024 22:30	C source file	1 KB
code_O1.o	19/04/2024 23:01	O File	1 KB
code_O1.s	19/04/2024 23:01	S File	1 KB
code_O2.c	19/04/2024 22:30	C source file	1 KB
code_O2.o	19/04/2024 23:01	O File	1 KB
code_O2.s	19/04/2024 23:01	S File	1 KB
code_O3.c	19/04/2024 22:30	C source file	1 KB
code_O3.o	19/04/2024 23:01	O File	1 KB
code_O3.s	19/04/2024 23:01	S File	1 KB
code_Ofast.c	19/04/2024 22:30	C source file	1 KB
code_Ofast.o	19/04/2024 23:01	O File	1 KB
code_Ofast.s	19/04/2024 23:01	S File	1 KB
code_Os.c	19/04/2024 22:30	C source file	1 KB
code_Os.o	19/04/2024 23:01	O File	1 KB
code_Os.s	19/04/2024 23:01	S File	1 KB

3. Perbandingan code\_O0.c, code\_O1.c, code\_O2.c, code\_O3.c, code\_Os.c, dan code\_Ofast.c. pada Notepad++

➤ code\_O0

```

code_O0.s
code_O1.s
code_O2.s
code_O3.s
code_Os.s
code_Ofast.s

1
2 code_O0.o:      file format pe-x86-64
3
4
5 Disassembly of section .text:
6
7 0000000000000000 <main>:
8   0:  55                push    %rbp
9   1:  48 89 e5          mov     %rsp,%rbp
10  4:  48 83 ec 30       sub     $0x30,%rsp
11  8:  e8 00 00 00 00    callq  d <main+0xd>
12 d:  c7 45 fc 00 00 00 00 movl    $0x0,-0x4(%rbp)
13 14:  c7 45 f8 00 00 00 00 movl    $0x0,-0x8(%rbp)
14 1b:  eb 0a            jmp     27 <main+0x27>
15 1d:  8b 45 fc          mov     -0x4(%rbp),%eax
16 20:  01 45 f8          add     %eax,-0x8(%rbp)
17 23:  83 45 fc 01       addl    $0x1,-0x4(%rbp)
18 27:  81 7d fc f3 01 00 00 cmpl    $0x1f3,-0x4(%rbp)
19 2e:  7e ed            jle     1d <main+0x1d>
20 30:  8b 45 f8          mov     -0x8(%rbp),%eax
21 33:  48 83 c4 30       add     $0x30,%rsp
22 37:  5d                pop     %rbp
23 38:  c3                retq
24 39:  90                nop
25 3a:  90                nop
26 3b:  90                nop
27 3c:  90                nop
28 3d:  90                nop
29 3e:  90                nop
30 3f:  90                nop
31

```

## ➤ code\_O1

```
code_O0.s x code_O1.s x code_O2.s x code_O3.s x code_Os.s x code_Ofast.s x
1
2 code_O1.o: file format pe-x86-64
3
4
5 Disassembly of section .text:
6
7 0000000000000000 <main>:
8 0: 48 83 ec 28 sub $0x28,%rsp
9 4: e8 00 00 00 00 callq 9 <main+0x9>
10 9: b8 f4 01 00 00 mov $0x1f4,%eax
11 e: 83 e8 01 sub $0x1,%eax
12 11: 75 fb jne e <main+0xe>
13 13: b8 4e e7 01 00 mov $0x1e74e,%eax
14 18: 48 83 c4 28 add $0x28,%rsp
15 1c: c3 retq
16 1d: 90 nop
17 1e: 90 nop
18 1f: 90 nop
19
```

## ➤ code\_O2

```
code_O0.s x code_O1.s x code_O2.s x code_O3.s x code_Os.s x code_Ofast.s x
1
2 code_O2.o: file format pe-x86-64
3
4
5 Disassembly of section .text.startup:
6
7 0000000000000000 <main>:
8 0: 48 83 ec 28 sub $0x28,%rsp
9 4: e8 00 00 00 00 callq 9 <main+0x9>
10 9: b8 4e e7 01 00 mov $0x1e74e,%eax
11 e: 48 83 c4 28 add $0x28,%rsp
12 12: c3 retq
13 13: 90 nop
14 14: 90 nop
15 15: 90 nop
16 16: 90 nop
17 17: 90 nop
18 18: 90 nop
19 19: 90 nop
20 1a: 90 nop
21 1b: 90 nop
22 1c: 90 nop
23 1d: 90 nop
24 1e: 90 nop
25 1f: 90 nop
26
```

## ➤ code\_O3

```

code_O0.s x code_O1.s x code_O2.s x code_O3.s x code_Os.s x code_Ofast.s x
1
2 code_O3.o: file format pe-x86-64
3
4
5 Disassembly of section .text.startup:
6
7 0000000000000000 <main>:
8 0: 48 83 ec 28 sub $0x28,%rsp
9 4: e8 00 00 00 00 callq 9 <main+0x9>
10 9: b8 4e e7 01 00 mov $0x1e74e,%eax
11 e: 48 83 c4 28 add $0x28,%rsp
12 12: c3 retq
13 13: 90 nop
14 14: 90 nop
15 15: 90 nop
16 16: 90 nop
17 17: 90 nop
18 18: 90 nop
19 19: 90 nop
20 1a: 90 nop
21 1b: 90 nop
22 1c: 90 nop
23 1d: 90 nop
24 1e: 90 nop
25 1f: 90 nop
26

```

## ➤ code\_Os

```

code_O0.s x code_O1.s x code_O2.s x code_O3.s x code_Os.s x code_Ofast.s x
1
2 code_Os.o: file format pe-x86-64
3
4
5 Disassembly of section .text.startup:
6
7 0000000000000000 <main>:
8 0: 48 83 ec 28 sub $0x28,%rsp
9 4: e8 00 00 00 00 callq 9 <main+0x9>
10 9: b8 4e e7 01 00 mov $0x1e74e,%eax
11 e: 48 83 c4 28 add $0x28,%rsp
12 12: c3 retq
13 13: 90 nop
14 14: 90 nop
15 15: 90 nop
16 16: 90 nop
17 17: 90 nop
18 18: 90 nop
19 19: 90 nop
20 1a: 90 nop
21 1b: 90 nop
22 1c: 90 nop
23 1d: 90 nop
24 1e: 90 nop
25 1f: 90 nop
26

```

## ➤ code\_Ofast

```
code_O0.s x code_O1.s x code_O2.s x code_O3.s x code_Os.s x code_Ofast.s x
1
2 code_Ofast.o: file format pe-x86-64
3
4
5 Disassembly of section .text.startup:
6
7 0000000000000000 <main>:
8 0: 48 83 ec 28 sub $0x28,%rsp
9 4: e8 00 00 00 00 callq 9 <main+0x9>
10 9: b8 4e e7 01 00 mov $0x1e74e,%eax
11 e: 48 83 c4 28 add $0x28,%rsp
12 12: c3 retq
13 13: 90 nop
14 14: 90 nop
15 15: 90 nop
16 16: 90 nop
17 17: 90 nop
18 18: 90 nop
19 19: 90 nop
20 1a: 90 nop
21 1b: 90 nop
22 1c: 90 nop
23 1d: 90 nop
24 1e: 90 nop
25 1f: 90 nop
26
```

## • Tugas 5 : Kompilasi Beberapa File Kode dengan GCC

### 1. Source Code dan Tampilan pada Notepad++

```
// Organisasi dan Arsitektur Komputer Tekom Kelas 2C
// Modul : 1
// Percobaan : 1
// Tanggal : 19 April 2024
// Nama (NIM) : Akbar Dwi Herlambang (2308979)
// Nama File : main_text.c
// Deskripsi : Kompilasi Beberapa File Kode dengan GCC

#include "text.h"
void main(void)
{
    test();
}
```

```
main_text.c
1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul : 1
3 // Percobaan : 1
4 // Tanggal : 19 April 2024
5 // Nama (NIM) : Akbar Dwi Herlambang (2308979)
6 // Nama File : main_text.c
7 // Deskripsi : Kompilasi Beberapa File Kode dengan GCC
8
9 #include "text.h"
10 void main(void)
11 {
12     test();
13 }
14
```

## 2. Tampilan text.c dan text.h beserta Source Codenya

### ➤ text.c

```
// Organisasi dan Arsitektur Komputer Tekom Kelas 2C
// Modul : 1
// Percobaan : 1
// Tanggal : 19 April 2024
// Nama (NIM) : Akbar Dwi Herlambang (2308979)
// Nama File : text.c
// Deskripsi : Proses Kompilasi Bahasa C Menggunakan GCC

#include <stdio.h>
#include "text.h"

void test(void)
{
    printf("Arsitektur Sistem Komputer sangat menyenangkan!\n");
}
```

```
main_text.c text.c text.h
1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul : 1
3 // Percobaan : 1
4 // Tanggal : 19 April 2024
5 // Nama (NIM) : Akbar Dwi Herlambang (2308979)
6 // Nama File : text.c
7 // Deskripsi : Proses Kompilasi Bahasa C Menggunakan GCC
8
9 #include <stdio.h>
10 #include "text.h"
11
12 void test(void)
13 {
14     printf("Arsitektur Sistem Komputer sangat menyenangkan!\n");
15 }
```

### ➤ text.h

```
// Organisasi dan Arsitektur Komputer Tekom Kelas 2C
// Modul : 1
// Percobaan : 1
// Tanggal : 19 April 2024
```

```
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : text.h
// Deskripsi       : Proses Kompilasi Bahasa C Menggunakan GCC

#ifndef TES_H
#define TES_H 100

    void test(void);

#endif
```

```
1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul      : 1
3 // Percobaan   : 1
4 // Tanggal    : 19 April 2024
5 // Nama (NIM)  : Akbar Dwi Herlambang (2308979)
6 // Nama File   : text.h
7 // Deskripsi   : Proses Kompilasi Bahasa C Menggunakan GCC
8
9 #ifndef TES_H
10 #define TES_H 100
11
12     void test(void);
13
14 #endif
15
```

### ➤ Tampilan kompilasi dan main\_text.exe

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

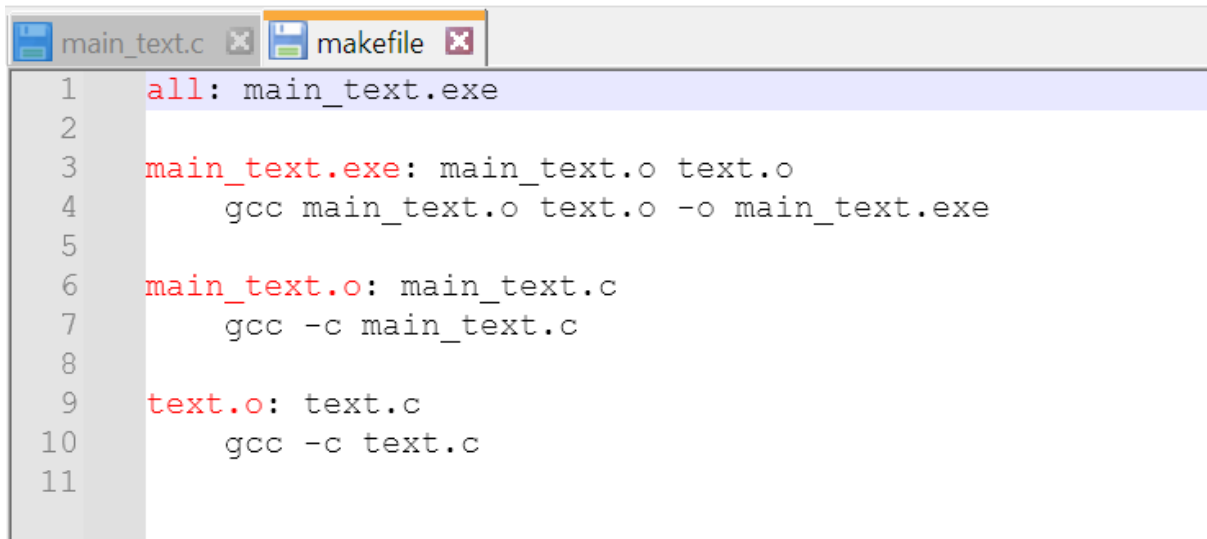
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 5>gcc -o main_text.exe text.c main_text.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 5>main_text.exe
Arsitektur Sistem Komputer sangat menyenangkan!

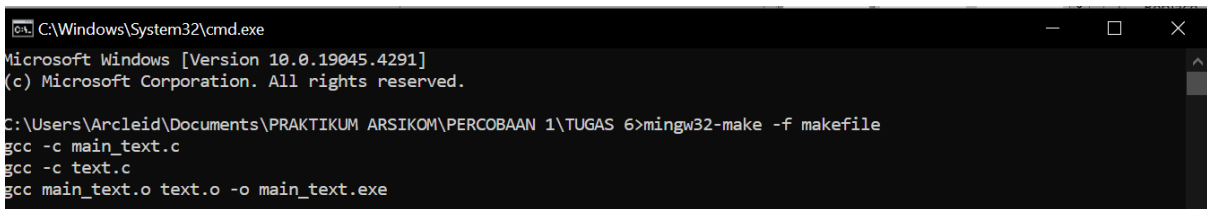
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 5>
```

## • Tugas 6 : Penggunaan Makefile pada GCC

### 1. Tampilan makefile dan eksekusi makefile



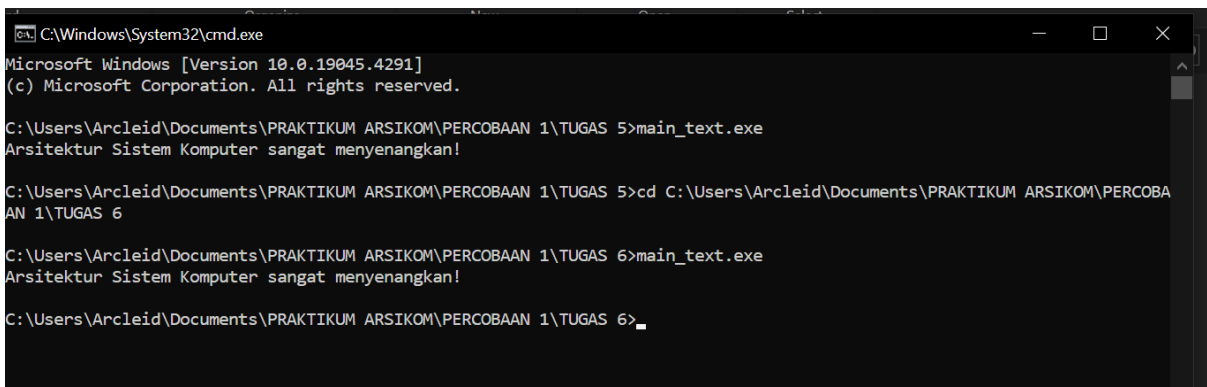
```
1 all: main_text.exe
2
3 main_text.exe: main_text.o text.o
4     gcc main_text.o text.o -o main_text.exe
5
6 main_text.o: main_text.c
7     gcc -c main_text.c
8
9 text.o: text.c
10    gcc -c text.c
11
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 6>mingw32-make -f makefile
gcc -c main_text.c
gcc -c text.c
gcc main_text.o text.o -o main_text.exe
```

## 2. Perbandingan hasil eksekusi main\_text.exe pada Tugas 5 dan 6



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 5>main_text.exe
Arsitektur Sistem Komputer sangat menyenangkan!

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 5>cd C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 6

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 6>main_text.exe
Arsitektur Sistem Komputer sangat menyenangkan!

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 6>_
```

- **Tugas 7 : Header File**

### 1. Tampilan add.h, add.c, dan main.c serta eksekusi dari main.c pada Command Prompt

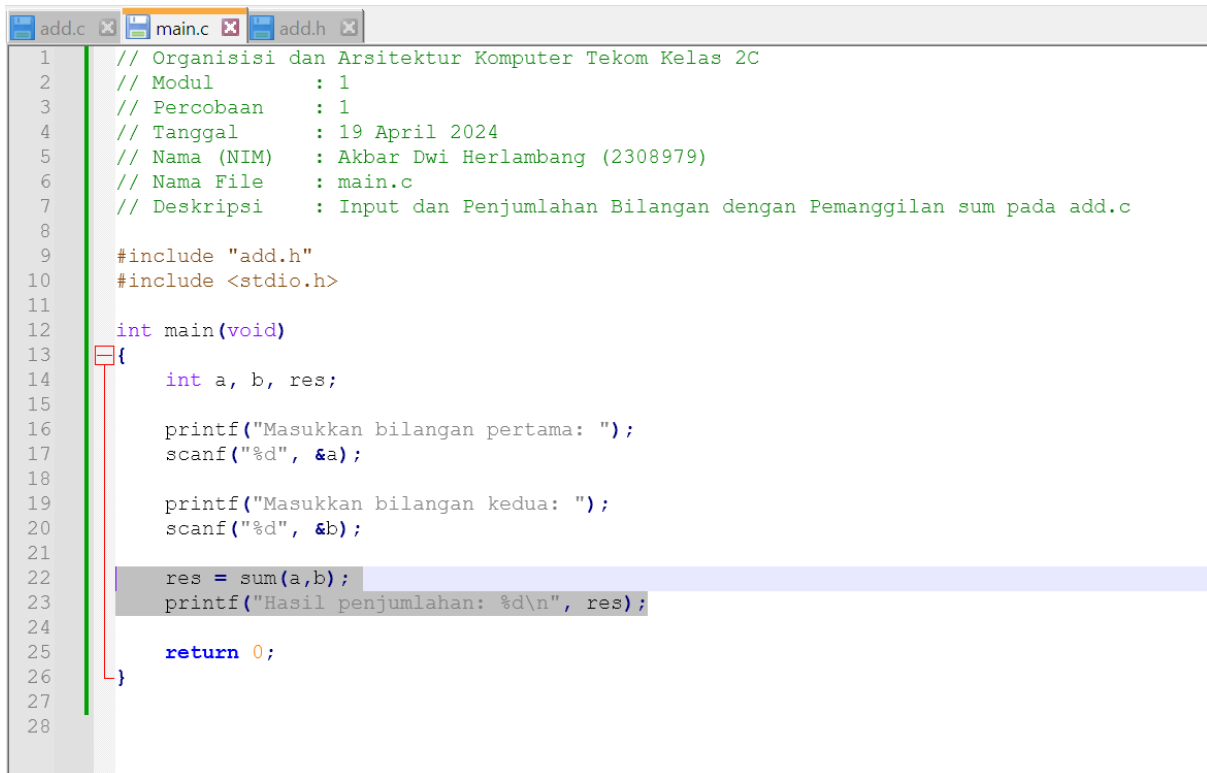
➤ add.h

```
add.c x main.c x add.h x
1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul : 1
3 // Percobaan : 1
4 // Tanggal : 19 April 2024
5 // Nama (NIM) : Akbar Dwi Herlambang (2308979)
6 // Nama File : add.h
7 // Deskripsi : Header File add.h yang akan di include pada main.c
8
9 #ifndef ADD_H
10 #define ADD_H
11
12 extern int accum;
13
14 int sum(int x, int y);
15
16 #endif
17
```

➤ add.c

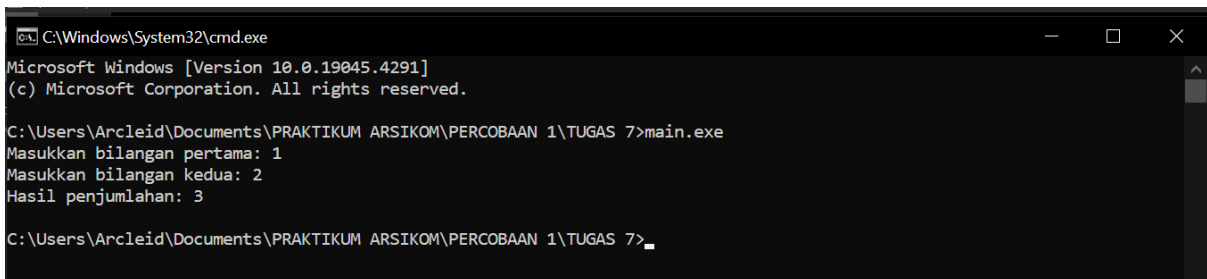
```
add.c x main.c x add.h x
1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul : 1
3 // Percobaan : 1
4 // Tanggal : 19 April 2024
5 // Nama (NIM) : Akbar Dwi Herlambang (2308979)
6 // Nama File : add.c
7 // Deskripsi : Header File
8
9
10 #define START_VAL 0
11 #include "add.h"
12
13 int accum = START_VAL;
14 int sum(int x, int y)
15 {
16     int t = x + y;
17     accum += t;
18     return t;
19 }
20
```

➤ main.c



```
1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul : 1
3 // Percobaan : 1
4 // Tanggal : 19 April 2024
5 // Nama (NIM) : Akbar Dwi Herlambang (2308979)
6 // Nama File : main.c
7 // Deskripsi : Input dan Penjumlahan Bilangan dengan Pemanggilan sum pada add.c
8
9 #include "add.h"
10 #include <stdio.h>
11
12 int main(void)
13 {
14     int a, b, res;
15
16     printf("Masukkan bilangan pertama: ");
17     scanf("%d", &a);
18
19     printf("Masukkan bilangan kedua: ");
20     scanf("%d", &b);
21
22     res = sum(a,b);
23     printf("Hasil penjumlahan: %d\n", res);
24
25     return 0;
26 }
```

### ➤ Eksekusi main.exe



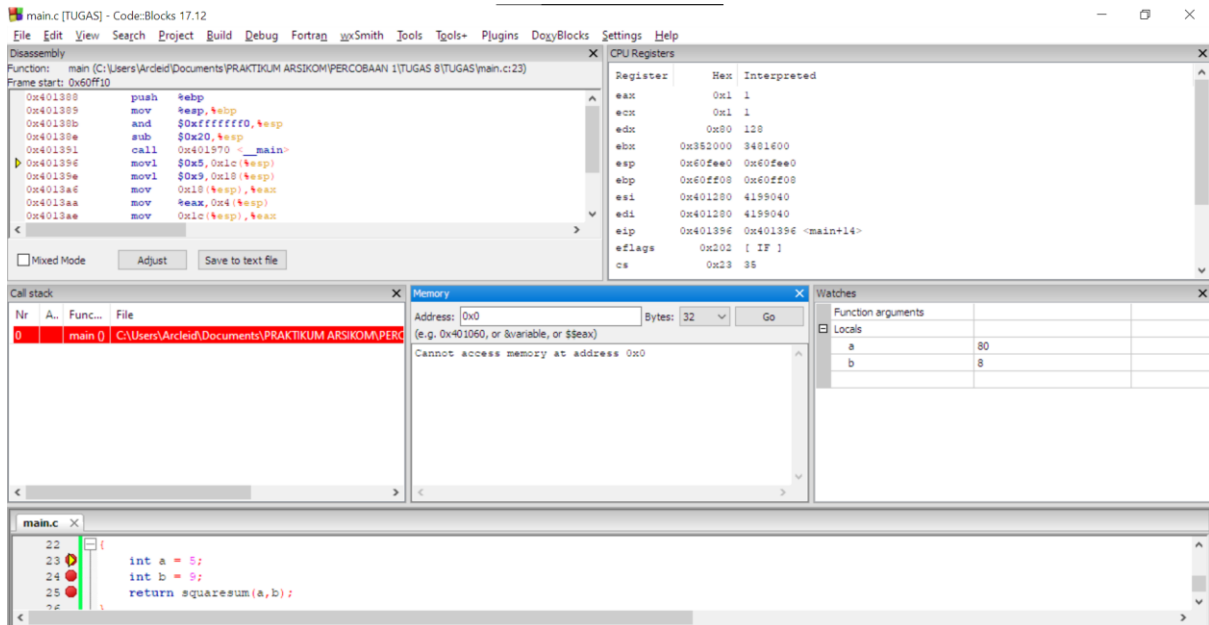
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 7>main.exe
Masukkan bilangan pertama: 1
Masukkan bilangan kedua: 2
Hasil penjumlahan: 3

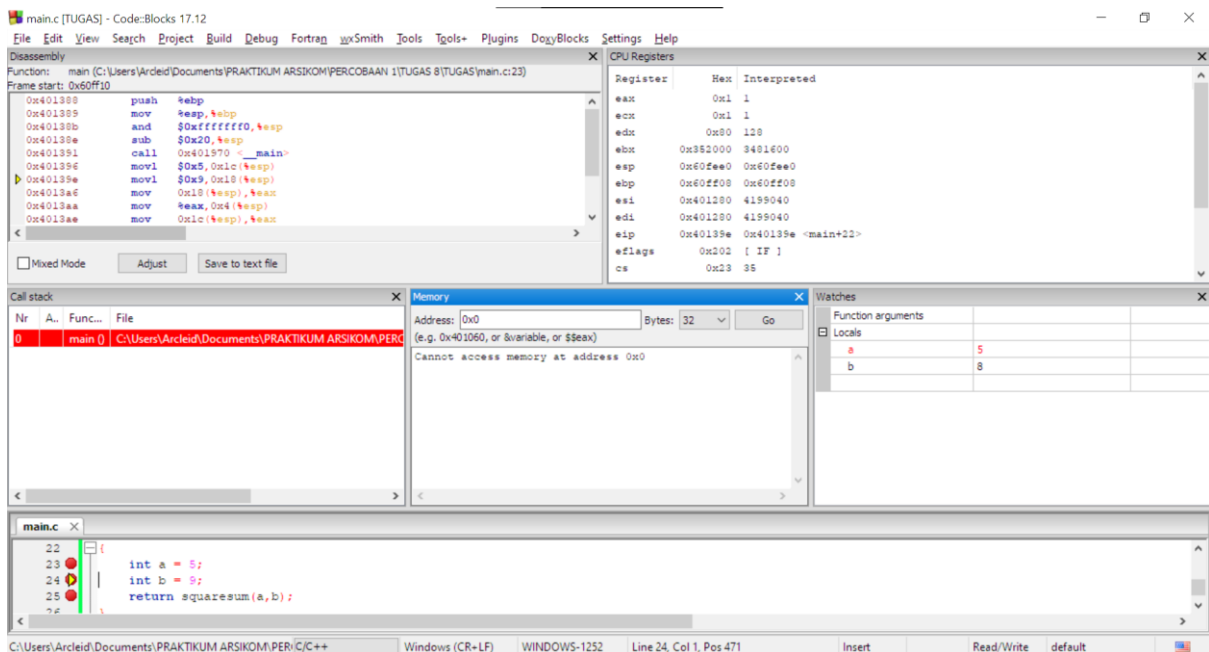
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 7>
```

## • Tugas 8 : Pemanggilan Prosedur dan Stack Memory

1. Tampilan debugging pada “int a = 5;”

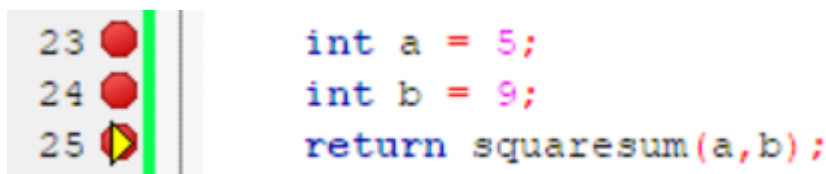


## 2. Tampilan debugging pada “int b = 9;”



## 3. Tampilan debugging pada “return squaresum(a,b)”, isi register ESP dan EBP, serta tampilan dari Memory Address ESP dan EBP.

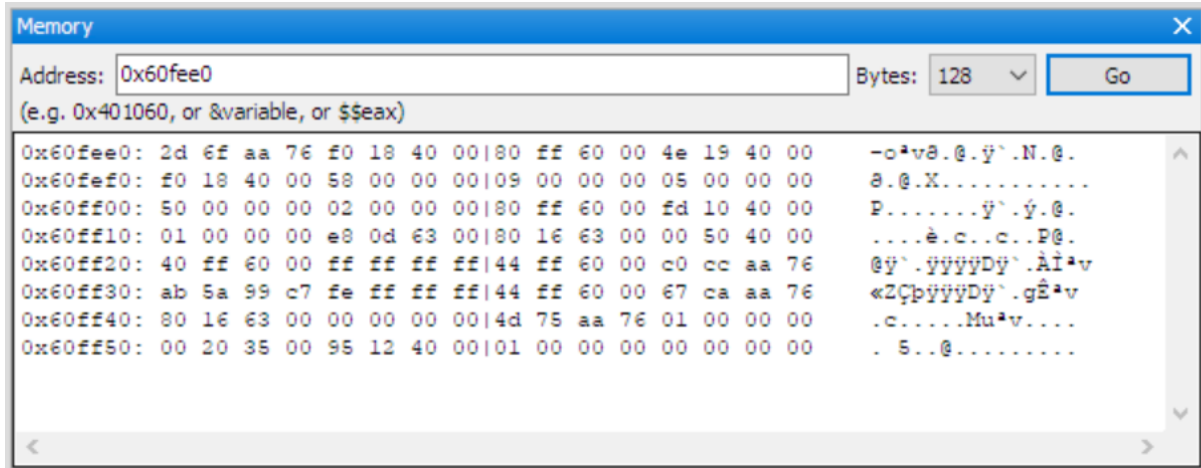
- Debug berada pada “return squaresum(a,b);”



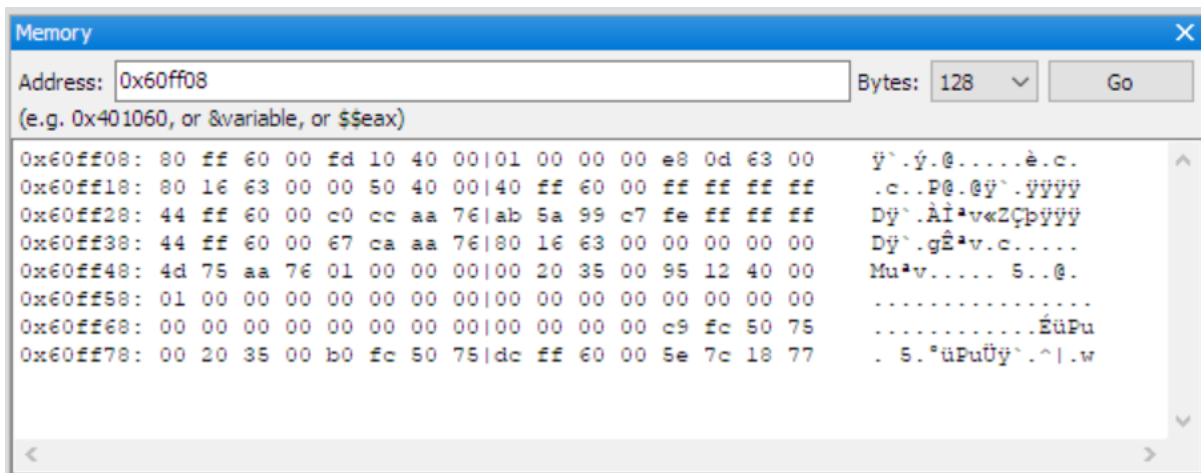
- Nilai ESP dan EBP

```
esp      0x60fee0  0x60fee0
ebp      0x60ff08  0x60ff08
```

- Memory Address ESP 128 Bytes



- Memory Address EBP 128 Byte



- Apa yang terjadi pada stack dalam proses diatas?

Dapat dilihat bahwa prosedur sudah masuk ke fungsi 'main' berdasarkan "return squaresum(a,b);" dan perintah sebelumnya yang mana "int a = 5;" dan "int b = 9;" ditambah dengan debug yang ditunjukkan pada *disassembly* yaitu "mov 0x18(%esp), %eax" yang berarti bahwa apa yang terjadi pada stack pada saat itu adalah, "0x18" berisikan nilai 'b' diambil dari stack lalu dipindahkan "mov" ke EAX.

4. Tampilan debugging pada "int temp1 = square(y);", isi register ESP dan EBP, serta tampilan dari Memory Address ESP dan EBP.

- Debug berada pada "int temp1 = square(y);"

```

15 {
16     int temp1 = square(y);
17     int temp2 = square(z);
18     return temp1+temp2;
19 }

```

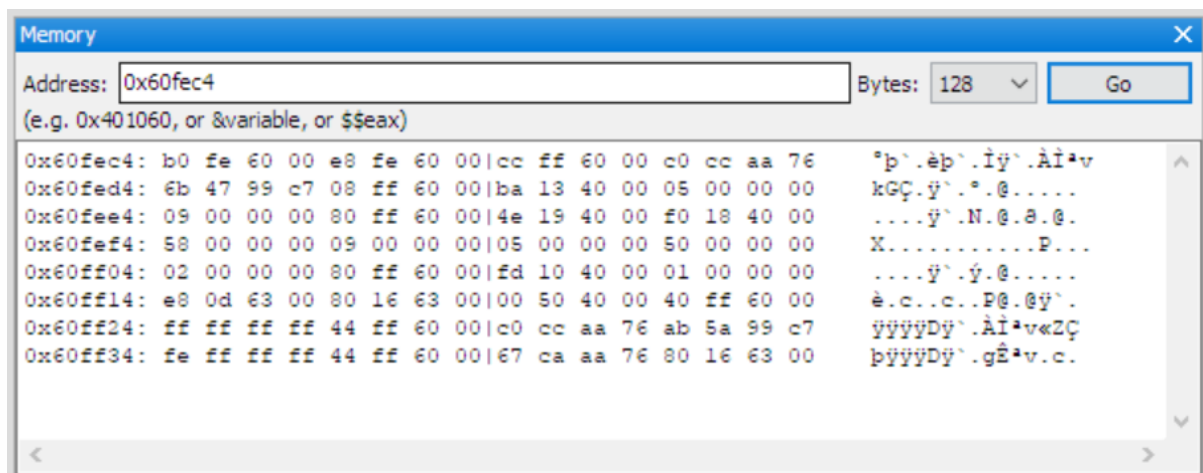
- Nilai ESP dan EBP

```

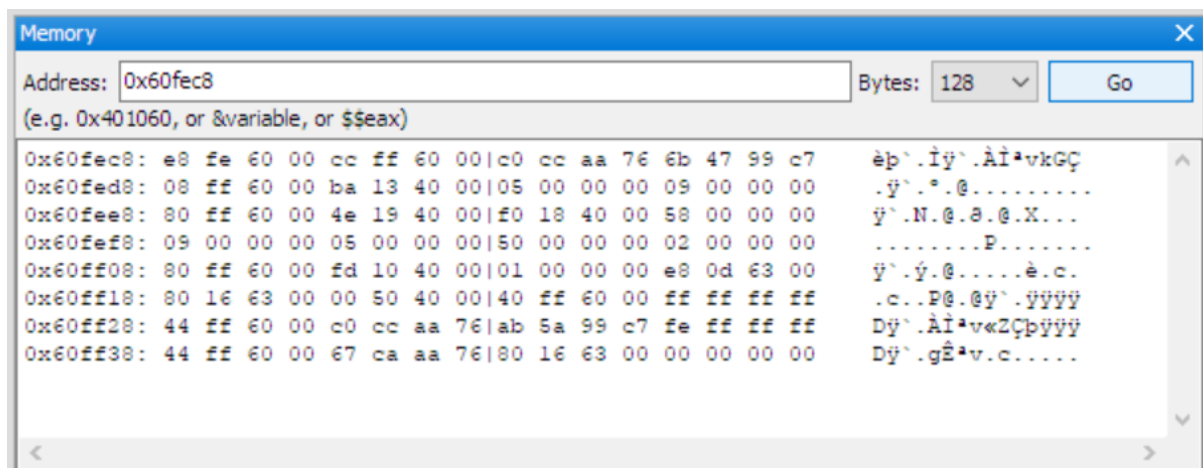
esp      0x60fec4  0x60fec4
ebp      0x60fed8  0x60fed8

```

- Memory Address ESP 128 Bytes



- Memory Address EBP 128 Byte



- Apa yang terjadi pada stack dalam proses diatas?

Dilihat dari debug *disassembly* yang mana saat ini berada pada “mov 0x8(%ebp),%eax” dan instruksi nya yaitu “int temp1 = square(y);” dapat disimpulkan bahwa yang terjadi pada stack adalah, memindahkan nilai offset 8

byte dari alamat awal EBP pada stack dan menyimpan/menyalinnya kedalam EAX.

5. Tampilan debugging pada “return x\*x;” isi register ESP dan EBP, serta tampilan dari Memory Address ESP dan EBP.

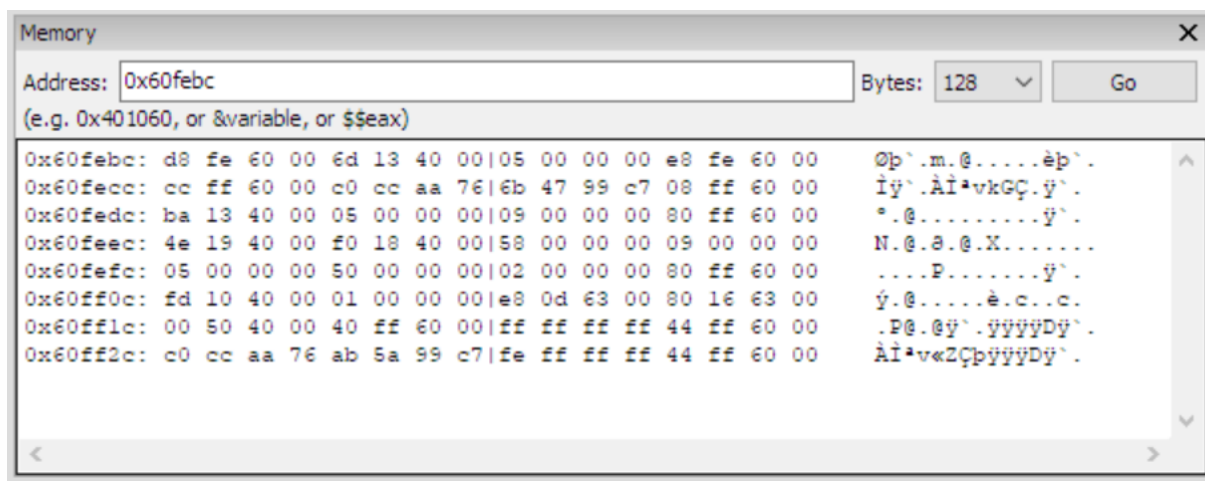
➤ Debug berada pada “return x\*x;”

```
9      int square (int x)
10     {
11     return x*x;
12     }
```

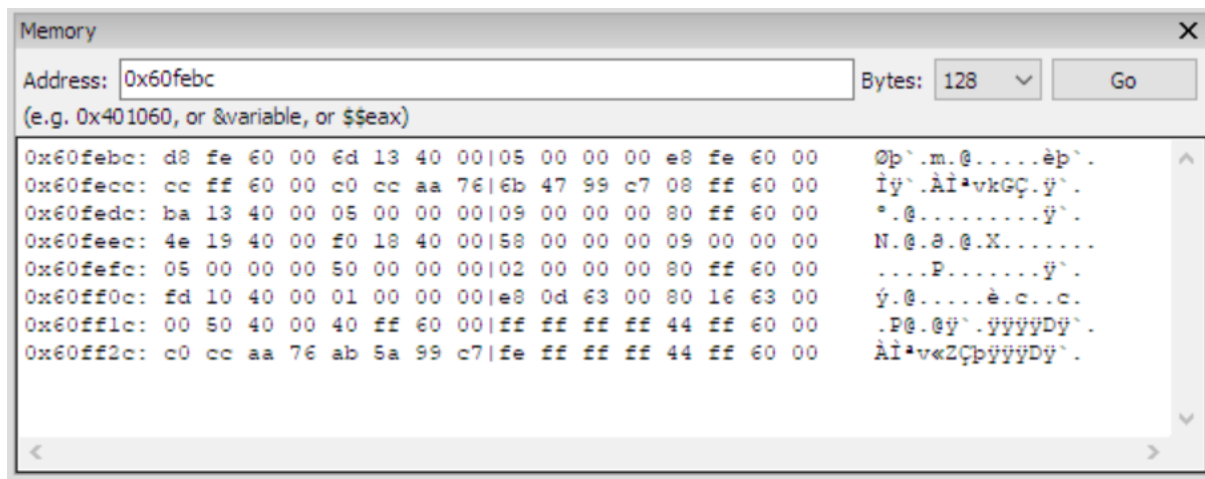
➤ Nilai ESP dan EBP

```
esp      0x60febc  0x60febc
ebp      0x60febc  0x60febc
```

➤ Memory Address ESP 128 Bytes



➤ Memory Address EBP 128 Bytes



➤ Call Stack

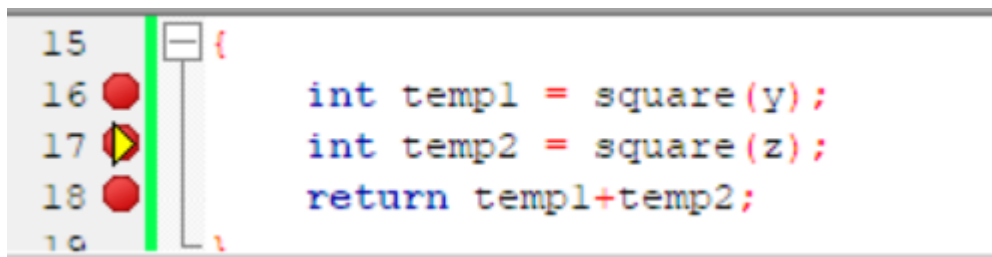
Call stack				X	
Nr	Address	Function	File		
0		square (x=5)	C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 8\TUGAS\main.c		
1	0x40136d	squaresum(y=5, z=9)	C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 8\TUGAS\main.c		
2	0x4013ba	main()	C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 8\TUGAS\main.c		

➤ Apa yang terjadi pada stack dalam proses diatas?

Dilihat dari debug *disassembly* yaitu “mov 0x8(%ebp),%eax” dan dengan fungsinya “ return x\*x;” kesimpulan pada apa yang terjadi di stack adalah, mengambil nilai offset 0x8 pada EBP lalu memindahkannya ke EAX.

6. Tampilan debugging pada “int temp2 = square(z);” isi register ESP dan EBP, serta tampilan dari Memory Address ESP dan EBP.

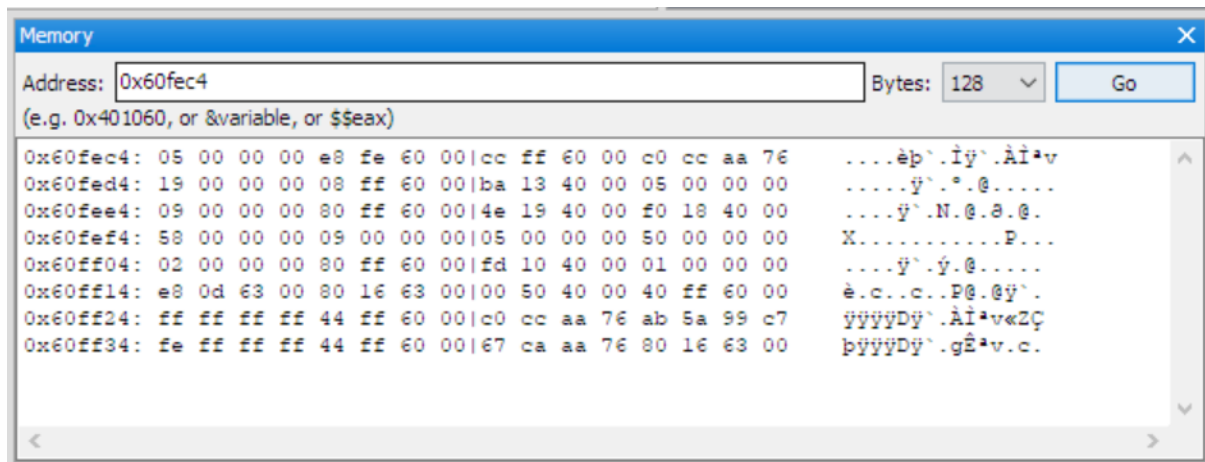
➤ Debug berada pada “int temp2 = square(z);”



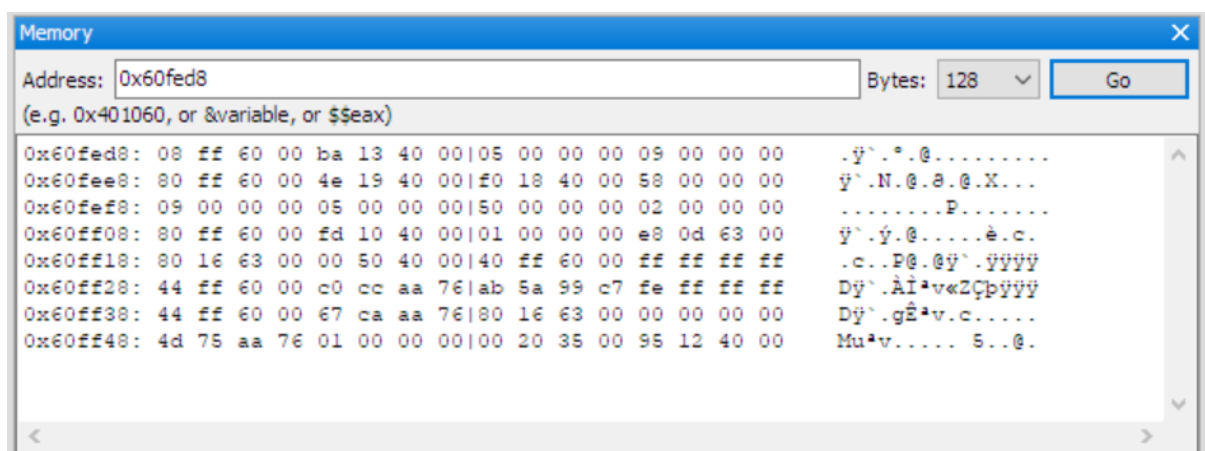
➤ Nilai ESP dan EBP

esp	0x60fec4	0x60fec4
ebp	0x60fed8	0x60fed8

➤ Memory Address ESP 128 Bytes



➤ Memory Address EBP 128 Bytes



➤ Call Stack

Call stack				
Nr	Address	Function	File	Line
0		squaresum (y=5, z=9)	C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 8\TUGAS\main.c	17
1	0x4013ba	main()	C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 8\TUGAS\main.c	25

➤ Apa yang terjadi pada stack dalam proses diatas?

Dilihat dari debug *disassembly* yaitu “mov 0xc(%ebp),%eax” dan dengan fungsinya “int temp2 = square(z);” dapat disimpulkan bahwa apa yang terjadi pada stack adalah, nilai base pointer EBP dengan offset 0xc diambil dan dipindahkan ke EAX

7. Tampilan debugging pada “return temp1 + temp 2;” isi register ESP dan EBP, serta tampilan dari Memory Address ESP dan EBP.

- Debug berada pada “return temp1 + temp2;”

```

15  {
16  int temp1 = square(y);
17  int temp2 = square(z);
18  return temp1+temp2;

```

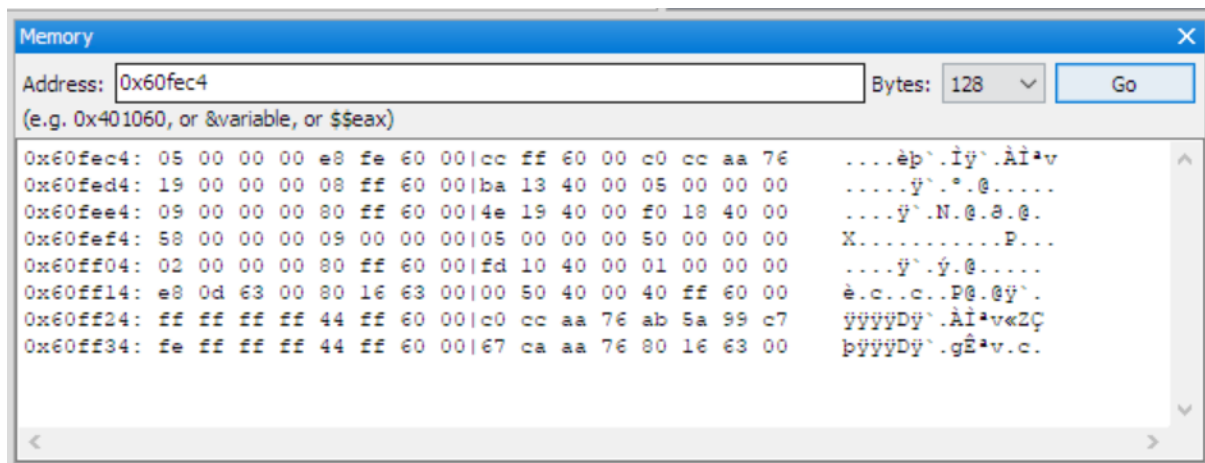
- Nilai ESP dan EBP

```

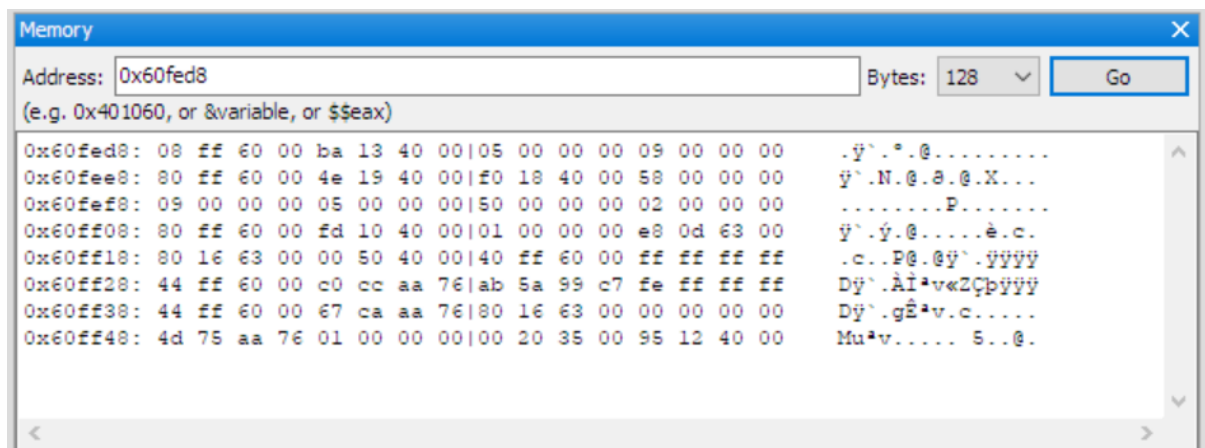
esp      0x60fec4  0x60fec4
ebp      0x60fed8  0x60fed8

```

- Memory Address ESP 128 Bytes



- Memory Address EBP 128 Bytes



- Apa yang terjadi pada stack dalam proses diatas?

Dilihat dari debug *disassembly* yaitu “mov 0xc(%ebp),%eax” dan dengan fungsinya “ return temp1+temp2; ”dapat disimpulkan bahwa apa yang terjadi pada stack adalah, mengambil -0x4 pada EBP dan menyimpannya ke EDX.

8. Tampilan debugging pada proses terakhir, isi register ESP dan EBP, serta tampilan dari Memory Address ESP dan EBP.

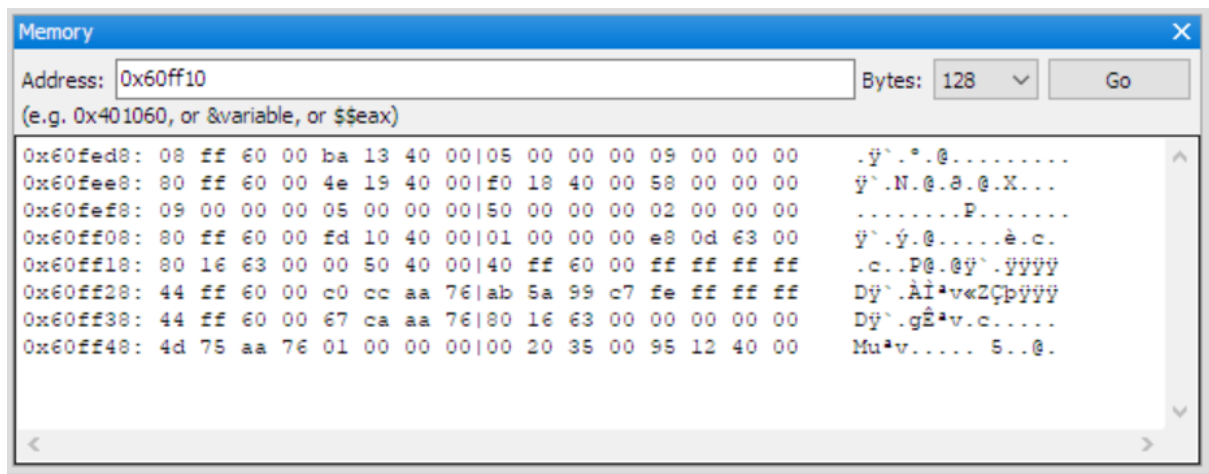
➤ Debug pada proses terakhir

```
23  int a = 5;
24  int b = 9;
25  return squaresum(a,b);
26  }
```

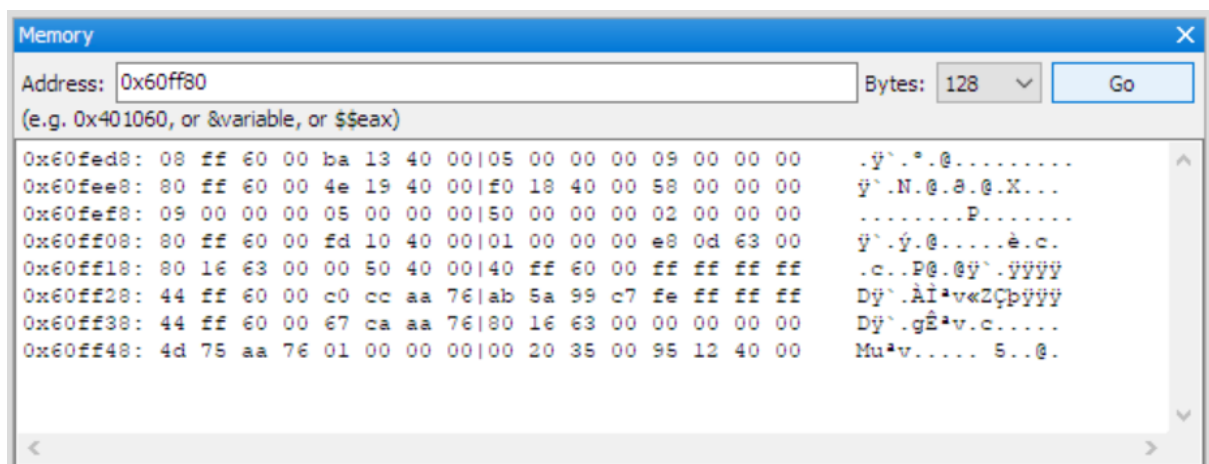
➤ Nilai ESP dan EBP

```
esp      0x60ff10  0x60ff10
ebp      0x60ff80  0x60ff80
```

➤ Memory Address ESP 128 Bytes



➤ Memory Address EBP 128 Bytes



➤ Call Stack

Call stack					
Nr	Address	Function	File	Line	
0	0x4010fd	__mingw_CRTStartup ()			
1	0x1	?? ()			
2	0x352000	?? ()			
3	0x77187c5e	ntdll!RtlGetAppContainerNamedObjectPath()	C:\Windows\SYSTEM32\ntdll.dll		
4	0x77187c2e	ntdll!RtlGetAppContainerNamedObjectPath()	C:\Windows\SYSTEM32\ntdll.dll		
5		?? ()			

➤ Apa yang terjadi pada stack dalam proses diatas?

Dilihat dari debug *disassembly* yaitu “mov %eax,%ebp” dapat disimpulkan bahwa apa yang terjadi pada stack adalah, memindahkan nilai EAX kembali ke EBP base pointer.

- **Tugas 9 : Program Fibonacci**

➤ Tampilan file inputn.c, Header File, Source Code, dan penjelasannya

```
// Organisasi dan Arsitektur Komputer Tekom Kelas 2C
// Modul      : 1
// Percobaan   : 1
// Tanggal      : 19 April 2024
// Nama (NIM)  : Akbar Dwi Herlambang (2308979)
// Nama File   : inputn.c
// Deskripsi    : Input untuk Fibonacci

#include <stdio.h>
#include "inputn.h"

int input_n()
{
    int a;
    do {
        printf("Masukkan angka (n >= 2): ");
        scanf("%d", &a);
        if (a < 2)
        {
            printf("Tolong Masukkan angka yang lebih besar daripada 2
\n");
        }
    } while (a < 2);

    return a;
}
```

```

1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul      : 1
3 // Percobaan   : 1
4 // Tanggal     : 19 April 2024
5 // Nama (NIM)  : Akbar Dwi Herlambang (2308979)
6 // Nama File   : inputn.c
7 // Deskripsi   : Input untuk Fibonacci
8
9 #include <stdio.h>
10 #include "inputn.h"
11
12 int input_n()
13 {
14     int a;
15     do {
16         printf("Masukkan angka (n >= 2): ");
17         scanf("%d", &a);
18         if (a < 2)
19         {
20             printf("Tolong Masukkan angka yang lebih besar daripada 2 \n");
21         }
22     } while (a < 2);
23
24     return a;
25 }
26

```

```

1 #ifndef INPUTN_H
2 #define INPUTN_H
3
4 int input_n();
5
6 #endif

```

Saya membuat Header berupa “inputn.h” seperti tugas 7, karena saya mencoba untuk langsung memasukkan “inputn.c” pada “fibonacci\_main.c” tanpa adanya Header File saya mengalami error terus menerus, maka dari itu saya mengikuti Langkah dari Tugas 7.

Program diatas berfungsi sebagai Input N dan Verifikasi bahwa angka yang diinput lebih daripada 2, menggunakan “do-while” sehingga ketika input kurang daripada 2 maka akan terus mengalami perulangan.

➤ Tampilan file fibo.c, Header File, Source Code, dan penjelasannya

```

// Organisasi dan Arsitektur Komputer Tekom Kelas 2C
// Modul      : 1
// Percobaan   : 1
// Tanggal     : 19 April 2024
// Nama (NIM)  : Akbar Dwi Herlambang (2308979)
// Nama File   : fibo.c
// Deskripsi   : Untuk Menghitung Fibonacci

#include <stdio.h>
#include "fibo.h"

void calc_fib(int n)
{
    int a = 0, b = 1, c;

    printf("Angka Fibonacci dari %d :\n", n);
    printf("%d, %d, ", a, b);
}

```

```

    for (int i = 3; i <= n; ++i)
    {
        c = a + b;
        printf("%d, ", c);
        a = b;
        b = c;
    }
    printf("\n");
}

```

```

1 // Organisasi dan Arsitektur Komputer Tekom Kelas 2C
2 // Modul : 1
3 // Percobaan : 1
4 // Tanggal : 19 April 2024
5 // Nama (NIM) : Akbar Dwi Herlambang (2308979)
6 // Nama File : fibo.c
7 // Deskripsi : Untuk Menghitung Fibonacci
8
9 #include <stdio.h>
10 #include "fibo.h"
11
12 void calc_fib(int n)
13 {
14     int a = 0, b = 1, c;
15
16     printf("Angka Fibonacci dari %d :\n", n);
17     printf("%d, %d, ", a, b);
18
19     for (int i = 3; i <= n; ++i)
20     {
21         c = a + b;
22         printf("%d, ", c);
23         a = b;
24         b = c;
25     }
26     printf("\n");
27 }
28

```

```

1 #ifndef FIBO_H
2 #define FIBO_H
3
4 void calc_fib(int n);
5
6 #endif

```

Sama seperti “inputn.c” saya mengalami kendala karena prosedur dari “fibo.c” tidak dapat dikenali dalam “fibo\_main.c” dan menggunakan Header File sebagai perbaikannya.

Program atau prosedur diatas berfungsi sebagai pembuat Deret Fibonacci berdasarkan dari input prosedur “inputn.c”, deret Fibonacci diawali oleh 0 dan 1, maka dari itu 2 angka pertama yang dikeluarkan dari variable a dan b. pada suku angka ke 3 akan masuk ke instruksi perulangan yang berawal dari perulangan ke-3 sampai dengan angka yang telah diinput.

➤ Tampilan file fibo\_main.c, Source Code, dan penjelasannya

```
// Organisasi dan Arsitektur Komputer Tekom Kelas 2C
```

```

// Modul      : 1
// Percobaan   : 1
// Tanggal      : 19 April 2024
// Nama (NIM)  : Akbar Dwi Herlambang (2308979)
// Nama File   : fibo_main.c
// Deskripsi    : Kompilasi dari inputn.c dan fibo.c

#include <stdio.h>
#include "inputn.h"
#include "fibo.h"

int main()
{
    int sum = input_n();
    calc_fib(sum);

    return 0;
}

```

Saya menggunakan Header File untuk identifikasi dari prosedur daripada file “.c” nya langsung, dan Program diatas berfungsi sebagai pemanggil dari kedua prosedur sebelumnya yaitu “input\_n” dan “calc\_fib”, karena “input\_n()” hanya mereturn nilai tanpa variable, maka nilai tersebut dimasukkan ke variable “sum” sebagai syarat untuk pemanggilan prosedur “calc\_fib(int n)”.

➤ Tampilan makefile, eksekusi makefile, dan contoh Fibonacci.exe

```

all: fibonacci.exe

fibonacci.exe: inputn.o fibo.o fibo_main.o
    gcc inputn.o fibo.o fibo_main.o -o fibonacci.exe


inputn.o: inputn.c
    gcc -c inputn.c

fibo.o: fibo.c
    gcc -c fibo.c

fibo_main.o: fibo_main.c

```

```
gcc -c fibo main.c
```



```
1 all: fibonacci.exe
2
3 fibonacci.exe: inputn.o fibo.o fibo_main.o
4 gcc inputn.o fibo.o fibo_main.o -o fibonacci.exe
5
6 inputn.o: inputn.c
7 gcc -c inputn.c
8
9 fibo.o: fibo.c
10 gcc -c fibo.c
11
12 fibo_main.o: fibo_main.c
13 gcc -c fibo_main.c
```

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 9>mingw32-make -f makefile
gcc -c inputn.c
gcc -c fibo.c
gcc -c fibo_main.c
gcc inputn.o fibo.o fibo_main.o -o fibonacci.exe

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 9>fibonacci.exe
Masukkan angka (n >= 2): 7
Angka Fibonacci dari 7 :
0, 1, 1, 2, 3, 5, 8,
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 1\TUGAS 9>
```

## • Simpulan

Kode sumber C diubah menjadi program executable: Kompilasi, pengompilan, dan linking proses di mana perangkat lunak pembuatan konverter compiler kode ke dalam kode mesin yang sesuai dengan arsitektur mikroprosesor target. File objek berisi kode mesin yang terbaca, dan hasil disassembly file .o dan .exe dapat berbeda, biasanya file .exe mengandung lebih banyak informasi. Opsi GCC untuk optimisasi c di mana kinerja dan efisiensi kode dipengaruhi, dan kecepatan eksekusi dapat diselesaikan . make file dapat mencapai tujuan otomatisasi proses kompilasi, header file membantu modularitas c dan pemeliharaan kode. Eksternal activar adalah deklarasi variabel diberikan dideklarasikan dalam file sumber. Dalam prosedur eksekusi, memori dipetakan ke stack untuk alokasi memori yang efisien dan pengelolaan otomatis. Tanggung jawab caller/pemanggil meliputi dari menyiapkan parameter dan lingkungan eksekusi, sementara callee bertanggung jawab untuk mengeksekusi fungsi dan mengatur memori lokal.