

**PRAKTIKUM 2**  
**IF2130 - Organisasi dan Arsitektur Komputer**  
**II2130 - Arsitektur dan Sistem Komputer**

**“PERMAINAN SOTONG”**  
**Reading Assembly**

Dipersiapkan oleh:  
Asisten Lab Sistem Terdistribusi

Didukung Oleh:



**Waktu Mulai:**  
Jumat, 29 Oktober 2021, 18.59.59 WIB

**Waktu Akhir:**  
Jumat, 5 November 2021, 18.59.59 WIB

## I. Latar Belakang



Anda adalah seorang pecandu gacha. Di dunia ini tiada satupun game gacha yang Anda lewati dan saat ini Anda sedang tertarik pada suatu game gacha (sebut saja G\*nsh\*n Imp\*\*t). Memainkan game tersebut setiap jam, setiap menit, setiap detik demi melakukan gacha. Mungkin saat ini bisa dikatakan bahwa gacha sudah menjadi bagian dari nafas Anda, mengalir di setiap jengkal dari jiwa dan raga. Mengapa bisa seperti ini? “Gacha menjanjikan kemenangan” itulah yang Anda percaya. Ya, benar. Kesenangan dari gacha memang memacu hormon - hormon dopamine dalam otak Anda, hormon kebahagiaan yang terasa bagaikan surga dunia dan candu.

Sayangnya janji-janji manis mengenai kemenangan dan kebahagiaan hanyalah sebatas isapan jempol belaka. Berkali-kali Anda mencoba dan mencoba untuk mencari kebahagiaan tersebut. Awalnya sekali sudah cukup bahagia, lama kelamaan sebarang apapun yang Anda dapatkan mulai terasa repetitif. Anda mencoba untuk mencari perasaan kebahagiaan tersebut dengan melontarkan lebih banyak uang, tetapi hanya ~~sekumpulan pedang~~ kegagalan yang Anda dapatkan. Tanpa sadar uang yang Anda gunakan sudah habis. Masih belum puas dengan segala yang didapatkan, Anda mulai meminjam uang dari perbankan untuk memenuhi hasrat tersebut. Lama kelamaan hidup Anda mulai tidak berkualitas.

Anda merasa lelah tanpa sebab di setiap waktu dan hanya menatap layar sepanjang hari. Debt collector mulai mendatangi rumah Anda dan menyita seluruh aset - aset yang ada, mereka mengancam akan mengambil organ tubuh Anda sebagai penggantinya jika tidak dapat memenuhi pembayaran selanjutnya. Sekarang yang Anda miliki hanyalah sekumpulan gambar 2 dimensi dari karakter yang Anda dapatkan dengan seluruh pengorbanan, tentu saja gambar-gambar karakter tersebut tidak dapat melunasi hutang Anda ataupun membuat hidup lebih baik.

Tanpa uang, tanpa atap, tanpa tujuan, Anda mulai berjalan ke tempat sekitar untuk melepas penat. Tanpa sengaja, Anda bertemu dengan seorang berbaju kantor. Ia menawarkan untuk berjudi dengannya. Karena Anda hanya memiliki gambar karakter 2D untuk dipertaruhkan, ia berjanji akan memberikan gambar kartun 2D jika Anda menang dan akan mengambil gambar - gambar kesayangan Anda jika kalah. Tanpa disangka, Anda memenangkan permainan tersebut pada percobaan pertama.

Ingatan - ingatan masa lalu kembali merasuki jiwa Anda, muncul perasaan yang serupa dengan memenangkan gacha. Adiksi kembali merasuki jiwa, Anda menginginkan lebih. Orang tersebut kemudian menawarkan Anda dalam sebuah perjalanan 4 hari 2 malam, Ia mengatakan bahwa Anda bisa mendapatkan lebih banyak karakter 2D disana. Dalam sekejap Anda menyetujui ajakan tersebut dan mulai masuk ke dalam sebuah mobil Van. Kelelahan bermain dan delusi tingkat tinggi membuat Anda tidur terlelap, tanpa sadar Anda sudah tiba di sebuah ruangan berisikan dengan pemain - pemain lain yang kehausan akan gambar 2D. Seseorang menggunakan baju merah mulai berbaris dan salah seorang dari mereka berkata

“Selamat datang di Permainan Sotong”

## II. Deskripsi Tugas

Pada praktikum ini, kalian akan mengeksplorasi cara kerja program dengan membaca dan memahami kode assembly. Tugas kalian adalah menebak input untuk fungsi-fungsi pada program yang diberikan serta menuliskan kode C yang sesuai dengan hasil disassembly. Dengan bantuan **gdb** atau tools sejenis kalian dapat memasang breakpoint, melihat isi register, perintah yang sedang dijalankan, hasil disassembly suatu fungsi, isi memory, serta informasi lain yang dapat membantu keberhasilan praktikum ini.

Perhatikan contoh disassembly gdb pada fungsi bernama contoh berikut.

Dump of assembler code for function contoh:

```
0x565561e5 <+0>:    push    %ebp
0x565561e6 <+1>:    mov     %esp,%ebp
0x565561e8 <+3>:    push    %ebx
0x565561e9 <+4>:    sub     $0x14,%esp
0x565561ec <+7>:    call    0x565560d0 <__x86.get_pc_thunk.bx>
0x565561f1 <+12>:   add     $0x2ddb,%ebx
0x565561f7 <+18>:   mov     %gs:0x14,%eax
0x565561fd <+24>:   mov     %eax,-0xc(%ebp)
0x56556200 <+27>:   xor     %eax,%eax
0x56556202 <+29>:   sub     $0x8,%esp
0x56556205 <+32>:   lea     -0x10(%ebp),%eax
0x56556208 <+35>:   push    %eax
0x56556209 <+36>:   lea     -0x1fc4(%ebx),%eax
0x5655620f <+42>:   push    %eax
=> 0x56556210 <+43>:   call    0x56556070 <__isoc99_scanf@plt>
0x56556215 <+48>:   add     $0x10,%esp
0x56556218 <+51>:   mov     -0x10(%ebp),%eax
0x5655621b <+54>:   cmp     $0x3,%eax
0x5655621e <+57>:   je      0x56556225 <contoh+64>
0x56556220 <+59>:   call    0x5655624e <trap>
0x56556225 <+64>:   sub     $0xc,%esp
0x56556228 <+67>:   lea     -0x1fc1(%ebx),%eax
0x5655622e <+73>:   push    %eax
0x5655622f <+74>:   call    0x56556030 <printf@plt>
0x56556234 <+79>:   add     $0x10,%esp
0x56556237 <+82>:   nop
```

```

0x56556238 <+83>:  mov    -0xc(%ebp),%eax
0x5655623b <+86>:  xor     %gs:0x14,%eax
0x56556242 <+93>:  je      0x56556249 <contoh+100>
0x56556244 <+95>:  call   0x565562f0 <__stack_chk_fail_local>
0x56556249 <+100>: mov     -0x4(%ebp),%ebx
0x5655624c <+103>: leave
0x5655624d <+104>: ret
End of assembler dump.

```

Hasil disassembly diatas didapatkan dengan menjalankan perintah **disassemble contoh**. Eksekusi program di atas dapat terhenti pada alamat 0x56556210 dikarenakan sebelumnya telah dijalankan perintah **break \*0x56556210**. Tanda panah (=>) sebelum instruksi menandakan bahwa baris instruksi tersebut adalah yang akan dieksekusi berikutnya. Berikut merupakan contoh penggunaan gdb yang perintahnya dapat kalian gunakan <https://asciinema.org/a/fmNC1VukUvEmfBsiLXCqn8ggx>

Berikut adalah cara membaca setiap baris instruksi hasil disassembly dengan contoh instruksi pada alamat 0x56556210.

Bagian	Makna
0x56556210 <+43>:	Alamat virtual instruksi (0x56556210) serta alamat relatif instruksi terhadap instruksi pertama pada fungsi contoh (+43).
call 0x56556070 <__isoc99_scanf@plt>	Instruksi assembly

Perhatikan potongan instruksi hasil disassembly dari alamat 0x56556210 hingga 0x56556225 berikut.

```

=> 0x56556210 <+43>:  call   0x56556070 <__isoc99_scanf@plt>
0x56556215 <+48>:  add     $0x10,%esp
0x56556218 <+51>:  mov     -0x10(%ebp),%eax
0x5655621b <+54>:  cmp     $0x3,%eax
0x5655621e <+57>:  je      0x56556225 <contoh+64>
0x56556220 <+59>:  call   0x5655624e <trap>
0x56556225 <+64>:  sub     $0xc,%esp

```

Pada potongan disassembly tersebut, program akan meminta input pada alamat 0x56556210 yang kemudian disimpan pada -0x10(%ebp). Pada alamat 0x56556218 input user akan dipindahkan ke register eax untuk kemudian dibandingkan dengan 0x3 pada

instruksi berikutnya. Jika sesuai, maka program akan melanjutkan instruksi pada alamat 0x56556225 sementara jika salah maka program akan memanggil fungsi trap. Dengan pemahaman tersebut, berikut merupakan contoh kode C yang sesuai dengan hasil disassembly fungsi contoh.

```
void contoh(){
    int a;
    scanf("%d", &a);

    if (a != 3) {
        illegal_move();
    }

    printf("Jawaban anda benar!");
}
```

Dengan demikian, kalian dapat memasukkan nilai 3 sebagai jawaban yang benar untuk mendapatkan poin dan menghindari trap. Setiap fungsi akan memiliki tantangannya sendiri, selamat mengerjakan!

## Langkah Umum Pengerjaan

1. Pastikan sistem operasi yang kalian gunakan adalah berbasis Linux. Kami **sangat menyarankan** menggunakan virtual machine. Alternatif lain adalah menggunakan Windows Subsystem for Linux 2 (**tapi dengan keterbatasan tertentu**) ataupun bisa dengan sistem dual boot (**metode riskan, dengan resiko yang harus ditanggung sendiri**). Kami merekomendasikan VirtualBox, dengan sistem operasi berbasis Debian seperti Ubuntu.  
VirtualBox dapat diunduh pada <https://www.virtualbox.org/wiki/Downloads>.  
File .iso Ubuntu dapat diunduh pada <https://ubuntu.com/>
2. Setelah sudah menyiapkan sistem operasi pada virtual machine, Anda diwajibkan untuk mengunduh GNU Project Debugger (gdb) pada sistem operasi kalian. Gunakan perintah (pada Ubuntu) `sudo apt install gdb` pada terminal Anda. Sesuaikan perintah dengan sistem operasi yang Anda pakai. Untuk menjalankan gdb, gunakan perintah `gdb nama_binary`.
3. Lakukan request teka-teki di <http://sotong.suggoi.fun/> dengan menggunakan token yang akan dikirimkan lewat email std.

4. Jalankan perintah `chmod +x nama_binary` pada terminal Anda di directory tempat Anda mengekstrak file hasil request. Hal ini dilakukan agar program dapat dijalankan.
5. Anda tidak disarankan untuk melakukan eksekusi file secara langsung. Anda dapat menggunakan perintah `gdb fifty-billion-game` pada terminal untuk menjalankan gdb.
6. Untuk menjalankan program dalam gdb, Anda dapat menggunakan perintah `run` dan memasukkan jawaban secara manual, maupun menggunakan file sebagai input. Misalnya Anda memasukkan jawaban dalam file `answer.txt` maka lakukan eksekusi dengan perintah `run answer.txt`. **Jangan asal melakukan run pada binary, karena apabila input Anda salah maka akan langsung mendapatkan nilai minus.**
7. Tips agar gdb lebih mudah dibaca yaitu unduh plugin berikut (salah satu saja sesuai keinginan Anda) yang membantu pewarnaan dan tata letak tulisan saat debugging:
  - a. <https://github.com/cyrus-and/gdb-dashboard>
  - b. <https://github.com/longld/peda>
  - c. <https://github.com/hugsy/gef>
  - d. <https://github.com/pwndbg/pwndbg>
8. Good luck, have fun!

## Penilaian

1. Setiap fungsi yang telah berhasil dijawab akan memberikan skor sebesar 10 atau 15 poin (tergantung kesulitan fungsi tersebut), serta setiap jebakan yang teraktifkan akan mengurangi nilai Anda sebesar 1 poin (berlaku kelipatan).

### III. Pengumpulan dan Deliverables

1. File teka-teki yang Anda gunakan bukanlah hasil kompilasi file `fifty-billion-game.c` sehingga jangan lakukan kompilasi terhadap file `fifty-billion-game.c`
2. Anda diwajibkan menulis kode C yang sesuai dengan hasil disassembly setiap fungsi. Fungsi yang berbeda dituliskan pada file `.c` yang berbeda, misalkan fungsi contoh dituliskan pada `contoh.c` dan fungsi `contoh_2` dituliskan pada `contoh_2.c`. Cukup menuliskan kode c untuk fungsi-fungsi berikut tanpa membuat fungsi main
  - a. Light
  - b. Sugar\_Honeycombs
  - c. Tug\_of\_War
  - d. Marbles, Glass, Squid (Pilih salah satu)
3. **Anda diwajibkan mengumpulkan answers.txt yang berisi jawaban beserta kode C dengan ketentuan pada poin 2.** Kumpulkan semua file dalam satu folder dengan NIM Anda sebagai nama folder. Kemudian, zip folder tersebut dengan nama yang sama. Berikut merupakan contoh struktur isi zip yang dikumpulkan.

```
13520420.zip
├── 13520420
│   ├── answers.txt
│   ├── Light.c
│   ├── Sugar_Honeycombs.c
│   ├── Tug_of_War.c
│   └── Marbles.c
```

4. Anda dapat melihat nilai yang tercatat pada server melalui scoreboard di <http://sotong.suggoi.fun/scoreboard>
5. **Mulai** Jumat, 29 Oktober 2021, 18.59 waktu server.  
**Deadline** Jumat, 5 November 2021, 18.59 waktu server.  
Server grading akan ditutup setelah waktu tersebut, menyebabkan pengerjaan yang melewati deadline tidak akan dinilai.
6. Dilarang melakukan serangan Denial of Service terhadap server.
7. Dilarang melakukan submisi dengan kode orang lain maupun men-submit dengan NIM orang lain. Kami memiliki rekap semua submisi yang anda lakukan sehingga segala bentuk kecurangan akan ditindak lanjuti.



8. Kami akan menindaklanjuti segala bentuk kecurangan yang terstruktur, masif, dan sistematis.
9. Diharapkan untuk mengerjakan sendiri terlebih dahulu sebelum mencari sumber inspirasi lain (Google, maupun teman anda yang sudah bisa). Percayalah jika menemukan sendiri jawabannya akan merasa bangga dan senang.
10. Dilarang melakukan kecurangan lain yang merugikan peserta mata kuliah IF2130 dan II2130 lainnya.
11. Perhatikan bagian **Latest Info** pada dashboard untuk informasi terbaru terkait dengan pengerjaan.
12. Jika ada pertanyaan atau masalah pengerjaan (atau ada yang tidak sengaja melakukan submit di luar format yang diberikan) harap langsung isi di [QnA](#). (Sister Tech Support)

## I.V Troubleshooting

### 4.1. No such file or directory

Jika OS 64 bit dan baru diinstal

1. Untuk menjalankan kode 32 bit dibutuhkan *library* versi 32 bit yang biasanya tidak otomatis diinstal. Jalankan kode berikut

```
sudo dpkg --add-architecture i386
sudo apt-get install libc6:i386 libstdc++6:i386
```

Jika masih gagal

1. Pastikan lokasi terminal di folder yang sama dengan file `fifty-billion-game`
2. Gunakan perintah `cd <folder>` untuk pindah ke folder tertentu atau lebih mudah klik kanan di file manager dan pilih `Open in Terminal`

### 4.2 Di GDB program *hang* saat dijalankan (*run*)

Jika saat perintah `run` diberikan program tidak bekerja (*stuck*) dan kadang muncul tulisan error seperti

```
warning: Breakpoint address adjusted from 0xf7fd9be0 to 0xfffffffff7fd9be0.
warning: Breakpoint address adjusted from 0xf7fda195 to 0xfffffffff7fda195.
warning: Breakpoint address adjusted from 0xf7fdbd1c to 0xfffffffff7fdbd1c.
warning: Breakpoint address adjusted from 0xf7fdb924 to 0xfffffffff7fdb924.
```

Kemungkinan besar disebabkan karena bug

<https://bugs.launchpad.net/ubuntu/+source/gdb/+bug/1848200>

Solusinya adalah menginstall GDB versi lebih lama dengan perintah berikut

```
sudo apt install gdb=8.1-0ubuntu3
```