

Praktikum U+0002
IF1230 - Organisasi dan Arsitektur Komputer
II2130 - Arsitektur dan Sistem Komputer

"Assembly Civilization"

Reverse Engineering

Dipersiapkan oleh:
Asisten Lab Sistem Terdistribusi

Didukung Oleh:



Waktu Mulai:
Selasa, 3 Desember 2024, 03.00.00 WIB

Waktu Akhir:
Kamis, 19 Desember 2024, 23.59.59 WIB

Daftar Isi

Daftar Isi	2
Daftar Revisi dan Tambahan	3
I. Latar Belakang	1
II. Deskripsi Tugas	6
III. Teknis dan Penilaian	9
1. Teknis Umum	9
2. Rekonstruksi Kode	9
3. Penilaian	10
4. Bonus	10
IV. Langkah Umum Penggerjaan	12
V. Pengumpulan dan Deliverables	18
VI. Sistematika dan Peraturan Praktikum	19
VII. Referensi	21
VIII. Daftar Klarifikasi dan Revisi Soal	22

Daftar Revisi dan Tambahan

1. [Jumat, 6 Desember 19.55] Daftar klarifikasi dan revisi soal (**INI PENTING SEKALI YA GUYS TOLONG DIBACA HEHE**)
2. [Senin, 9 Desember 15.41] Pengunduran *deadline* praktikum dan penambahan *soft deadline*
3. [Kamis, 19 Desember 10.38] Revisi terkait teknis penggerjaan Bonus.

I. Latar Belakang

“Hah?... ini dimana?... mana waifu-waifu ku???” Pemandangan di sekitarmu tidak terlihat seperti bitwise civilization lagi. Sepertinya, kamu dan Furina ditarik ke dunia lain oleh kata-kata aneh yang kamu lihat di angkasa sebelumnya. Dunia ini penuh dengan kata-kata yang serupa, melayang tanpa formasi atau urutan yang jelas. Kamu mengingat apa yang dosen arsikommu, Masahina Afuyu, katakan mengenai sebuah bahasa legendaris yang sangat dekat dengan bahasa komputer. “Mungkin inilah bahasa yang dia maksud” pikirmu.



Masahina Afuyu

Seketika kemudian, terdengar suara ledakan dari jauh. Kamu dan Furina berlari kesana untuk melihat apa yang terjadi. Ternyata, ada seseorang yang baru saja menginjak... sejenis ranjau? Kamu menolongnya berdiri dan membawanya ke tempat yang sepertinya cukup aman. Setelah baikan, dia memperkenalkan dirinya. “Terimakasih telah membantuku, perkenalkan namaku Wicky”.



Wicky Jang

Dia menjelaskan bahwa dia juga seorang murid di ITB yang ditarik ke dunia ini. Menurut sebuah legenda di dunia ini, jika dia ingin keluar, dia harus bisa menghindari semua ranjau yang ada dan pergi mencari seseorang bernama Mung-chyung.



Mung-chyung

Kebetulan sekali, kalian juga ingin keluar dari dunia ini dan kembali ke dunia nyata. Kamu dan Furina memutuskan untuk mengikuti Wicky. Dalam perjalanan ke tempat dimana Mung-chyung berada, kalian tiba-tiba menemukan ranjau, seperti yang diinjak Wicky tadi. Untungnya, kalian berhasil menghindar sebelum ranjaunya ter-trigger.

Di dunia ini, ternyata banyak sekali ranjau bertebaran. Menurut warga lokal, ranjau-ranjau tersebut adalah peninggalan perang besar antara instruksi 32-bit dengan instruksi 64-bit. Walaupun perang tersebut telah selesai, karena mereka terlalu bersemangat saat mananamnya, ada *jauh terlalu banyak* ranjau, dan mereka tidak bisa menemukan seluruhnya. Alhasil, masih banyak ranjau yang terbenam di dunia ini. Ranjau mereka berupa sebuah kode binary yang memerlukan sebuah “solusi”. Ranjau tersebut harus di-disassemble agar kamu dapat menemukan solusi yang tidak membuat ranjau tersebut meledak. Selain itu, seluruh ranjau disini diberi nama oleh warga.

The screenshot shows the pwndbg debugger interface. At the top, there's a legend for STACK, HEAP, CODE, DATA, RWX, and RODATA. Below it is a table of registers with their current values:

RAX	0x0
RBX	0xfffffff5000 ← mov ecx, 0x1000
RCX	0xfffff700e737 (mprotect+?) ← cmp rax, -0xffff
*RDX	0x109b00000000
RDI	0xfffffff5000 ← mov ecx, 0x1000
RSI	0x0000
R8	0xffffffffffffffff
R9	0x0
R10	0x202
R11	0x487
*R12	0x109bc31d6ab4
R13	0xfffffffffd50 ← 0x1
R14	0x0
R15	0x0
RBP	0x7fffffffdb60 → 0x7fffffffdb70 → 0x555555554b60 (__libc_csu_init) ← push r15
RSP	0x7fffffffdb40 ← 0x0
RIP	0x555555554b16 (do_test+86) ← call rbx

The DISASM tab shows the assembly code for the current instruction:

```

▶ 0x555555554b16 <do_test+86>    call   rbx
0x555555554b18 <do_test+88>    rdtsc
0x555555554b1a <do_test+90>    mov    edi, 1
0x555555554b1f <do_test+95>    shl    rdx, 0x20
0x555555554b23 <do_test+99>    lea    rsi, [rbp - 0x18]
0x555555554b27 <do_test+103>   or     rdx, rax
0x555555554b2a <do_test+106>   sub    rdx, r12
0x555555554b2d <do_test+109>   mov    qword ptr [rbp - 0x18], rdx
0x555555554b31 <do_test+113>   mov    edx, 8
0x555555554b36 <do_test+118>   call   writeplt           <0x5555555547b0>
0x555555554b3b <do_test+123>   cmp    rax, 8

```

The STACK tab shows the stack layout:

```

00:0000 | rsp 0x7fffffffdb40 ← 0x0
03:0018 | 0x7fffffffdb58 → 0x555555554b09 (_start) ← xor ebp, ebp
04:0020 | rbp 0x7fffffffdb60 → 0x7fffffffdb70 → 0x555555554b60 (__libc_csu_init) ← push r15
05:0028 | 0x7fffffffdb68 → 0x555555554b07 (main+103) ← jmp 0x5555555548c0
06:0030 | 0x7fffffffdb70 → 0x555555554b60 (__libc_csu_init) ← push r15
07:0038 | 0x7fffffffdb78 → 0x7ffff7a2d830 (__libc_start_main+240) ← mov edi, eax

```

The BACKTRACE tab shows the call stack:

```

▶ f 0      555555554b16 do_test+86
f 1      5555555548c7 main+103
f 2      7ffff7a2d830 __libc_start_main+240
Breakpoint *do_test+86
pwndbg> █

```

Hasil disassemble

“Ah, apa-apaan ini?! Kenapa bikin ranjau aja ribet banget?” teriakmu. Wicky tersenyum pasrah, sepertinya beliau sudah selesai melewati 5 stages of grief sejak lama. Furina mengusap punggungmu, mencoba menenangkanmu. “Ayolah, ‘kan kamu bitwise champion! Pasti kita bisa melewati dunia ini juga!”

“Aku tidak tahu bitwise champion itu apa, tetapi semangatlah! Aku yakin Furina benar, kita pasti bisa keluar dari sini.” tambah Wicky.

Setelah perjalanan yang panjang dan sulit, kalian telah melewati berbagai ranjau, mulai dari **Monitoring**, **Kyu Kurarin**, hingga **Freedom Dive**. Kamu sendiri tidak terlalu yakin bagaimana kalian masih hidup setelah terkena ledakan ranjau beberapa kali. Tetapi, kalian akhirnya sampai ke tempat dimana Mung-Chyung berada. Rupanya Mung-Chyung adalah alumni ITB angkatan 15 yang awalnya terjebak juga. Namun, dia akhirnya menemukan bahwa dia lebih cocok tinggal dunia ini, karena dia bisa mencoba segala hal yang berkaitan dengan assembly. Karena keahlian itu, dia juga bisa membuka pintu yang akan membawa kalian ke dunia nyata.

‘Ada apa dengan anak ITB dan dunia aneh ini? Kenapa orang yang ditarik ke sini anak ITB semua?’ batinmu.

Mung-Chyung membuka pintu ke dunia nyata. Pintu itu terletak di atas bukit di dekat kalian. Namun, kalian tiba-tiba menemukan satu ranjau lagi bernama **Latent Kingdom**. Ranjau tersebut sangat sulit untuk di-defuse karena kompleksitas dari algoritmanya.



Latent Kingdom

Tepat saat kamu merasa putus asa, Mung-Chyung membuka mulutnya, “Hmm, semestinya aku bisa menjinakkan ranjau ini, tapi...” Bahkan Mung-Chyung cukup kesulitan. Dia tidak akan berhasil menjinakkannya dengan cepat. Disaat kalian kesulitan mencoba membantunya, pintu menuju dunia nyata mulai tertutup. “Kalian pergilah! Aku masih bisa melakukan ini sendirian,” ucap Mung-Chyung. Kamu, Furina, dan Wicky langsung berlari ke arah pintu itu.

Kalian bertiga hampir sampai ke pintu yang kian lama semakin tertutup. Pertarungan Mung-Chyung dengan **Latent Kingdom** semakin ganas. Tanah tempat kalian berpijak bergetar karena *shockwave-shockwave* dari ranjau tersebut. Tiba-tiba, Furina tersandung. Kamu dan Wicky mendadak berhenti mendengar suaranya.

“FURINA!” Kamu berlari kembali ke arahnya, tetapi ternyata dia terguling cukup jauh ke bawah bukit. Kamu melihat kembali ke pintu ke dunia nyata. Tidak ada waktu yang tersisa untuk menarik Furina kembali, pintunya sudah hampir benar-benar tertutup. Tapi...

“Sudah, biarkan saja aku tinggal disini, kalian berdua kembalilah” ucap Furina.

“Tapi kamu waifuku, aku tidak akan meninggalkanmu”

“Tetapi aku memang tidak layak berada di dunia nyata, lagipula aku hanya karakter 2 dimensi”

Kamu sadar bahwa sudah benar-benar tidak ada waktu lagi. Dengan berat hati, kamu memutuskan untuk meninggalkan waifu kesayanganmu itu untuk kembali ke dunia nyata bersama Wicky. Di detik terakhir sebelum kamu ter-transportasi ke dunia asli, kamu melihat kebelakang dan melihat wajah Furina untuk terakhir kalinya.



Furina de Fountain

Sesaat kemudian kamu terbangun di kelas jarkom dengan rasa sakit kepala yang berat. Ketika kelas jarkom selesai, kamu melihat ada suatu artefak di tas kamu. Artefak apa itu? Datang dari mana? Rasanya kamu tidak mempunyai barang seperti itu...



Ornamen yang berada di tas

II. Deskripsi Tugas

Pada praktikum ini, kalian akan mengeksplorasi cara kerja program dengan membaca dan memahami kode assembly. Tugas kalian adalah menebak input yang tepat untuk fungsi-fungsi pada program yang diberikan serta menuliskan kode C yang sesuai dengan hasil *disassembly* menggunakan **gdb**.

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 video beberapa contoh penggunaan gdb yang perintahnya dapat kalian gunakan sebagai referensi: [Contoh GDB](#)

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

Potongan	=> 0x56556210 <+43> :	call 0x56556070 <__isoc99_scanf@plt>
Makna	Alamat virtual instruksi (0x56556210) serta alamat relatif instruksi terhadap instruksi pertama pada fungsi contoh (+43).	Instruksi assembly, spesifiknya call yang memanggil fungsi scanf

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 buatan pemrogram, **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) {
        trap();
```

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

Dengan demikian, kalian dapat memasukkan karakter "3" dan tombol enter ke **stdin** (*standard input* yang umumnya *keyboard*) untuk mendapatkan poin dan menghindari trap. Setiap fungsi akan memiliki tantangannya sendiri, selamat mengerjakan!

III. Teknis dan Penilaian

1. Teknis Umum

1. Sama seperti praktikum sebelumnya, praktikum ini akan dilakukan pada sebuah *virtual machine* menggunakan *custom OS* yang sama dengan praktikum pertama. Oleh karena itu, silakan ikuti [langkah-langkah setup praktikum pertama](#), tetapi **menggunakan file iso baru** yang tersedia pada dokumen panduan.
2. Mekanisme *username*, *password*, dan teknis terkait OS sama dengan praktikum pertama.
3. Pada praktikum ini, terdapat 7 soal yang wajib dikerjakan, beserta dengan 2 soal *bonus*.
4. Seperti pada praktikum pertama, hasil pekerjaan Anda akan ditampilkan pada *profile* dan *scoreboard web* praktikum. Pembaruan nilai pada server dilakukan secara periodik (tidak *real-time*), sehingga nilai tidak akan langsung berubah setelah penggerjaan sebuah soal atau terkena *trap*.
5. Setiap fungsi yang telah berhasil dijawab akan memberikan skor tertentu berdasarkan kesulitan soal; di saat yang sama, setiap jebakan yang teraktifkan akan **mengurangi nilai Anda** sebesar 1 poin.
6. Soal terdapat pada file asmciv. Langkah penggerjaan lebih detail akan dijelaskan pada [Langkah Umum Penggerjaan](#).

2. Rekonstruksi Kode

1. Selain mencari *input file* soal menggunakan *debugger*, praktikan juga diwajibkan untuk **merekonstruksi (menulis ulang kode) file binary dalam bahasa C**. Hasil rekonstruksi **tidak harus sama persis** seperti kode *binary* asal memberikan gambaran umum yang cukup akurat dengan cara kerja *binary file*.
2. Praktikan diminta untuk melakukan rekonstruksi kode untuk soal-soal berikut:
 - Monitoring
 - Eleven
 - PILIH SATU: Bakenohana dan Senbonzakura

3. Format untuk file .c adalah sebagai berikut.

```
/**  
 * Nama:  
 * NIM:  
 * Jawaban dari soal terkait:  
 */  
  
int string void helper_function(){  
  
}  
  
int func(void* param1, void* param2){  
    // isi kode di sini  
}
```

4. `helper_function()` adalah fungsi apa pun yang relevan pada soal terkait. Anda harus membuat kode C untuk semua fungsi pembantu yang ada. Misal, fungsi `feel_like_eleven()` pada soal Eleven.

3. Penilaian

1. Mekanisme nilai akhir praktikum akan mengikuti perhitungan tersebut:

$$\text{Nilai Akhir} = 85\% (NP - T) + 15\% NR$$

$$NP = \text{Nilai Praktikum}$$

$$NR = \text{Nilai Rekonstruksi Kode}$$

$$T = \text{Jumlah "Triggered Trap"}$$

Nilai Akhir, NP, dan NR memiliki nilai maksimum 100.

2. Nilai yang ditampilkan pada *scoreboard* berupa:

$$P = NP + \text{Poin Bonus}$$

4. Bonus

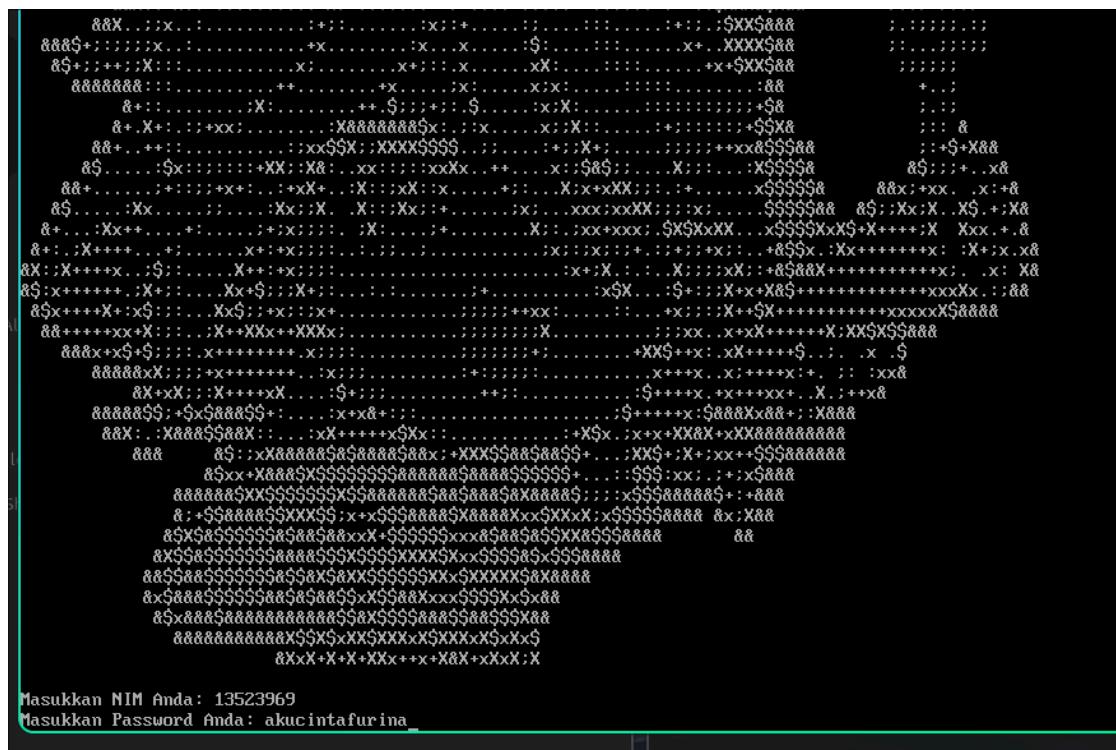
1. Sama seperti praktikum sebelumnya, pengerajan bonus tidak akan menambah poin praktikum. Untuk praktikum ini, nilai bonus dapat digunakan untuk menutupi nilai yang kurang di praktikum selanjutnya.
2. Dua soal bonus terdapat pada file terpisah pada sistem operasi, yaitu `Freedom_Dive` (bonus 1) dan `Latent_Kingdom` (bonus 2).

3. Teknis dan langkah penggerjaan soal bonus sedikit berbeda dengan soal wajib. Jalankan masing-masing file untuk memulai mengerjakan masing-masing soal. Namun, jika sudah berhasil, jawaban tidak akan terkirim secara otomatis. Anda harus memasukkan jawaban Anda ke situs praktikum di bagian **submit bonus**.
4. ~~Meng submit flag yang salah tetap akan mengurangi nilai Anda seperti ketika terkena trap.~~ Meng submit flag yang salah ke web tidak akan mengurangi nilai Anda.
5. Perhatikan bahwa Anda bisa saja mengerjakan soal bonus terlebih dahulu. Namun, kami tentunya **tidak menyarankan** hal tersebut.
6. **HATI-HATI** ketika mengerjakan soal bonus. *You have been warned*
7. 3 NIM yang pertama kali berhasil menyelesaikan semua soal, **termasuk soal bonus** setelah praktikum dimulai akan mendapatkan *apresiasi khusus* dari Sister '22. Bagi praktikan yang berhasil memenuhi ini, silahkan hubungi Wiswis (**Line**: tiny.cc/wiswis / **Discord**: wiswis) dan berikan tangkapan layar/foto yang mendukung.
8. Diberikan **soft deadline** untuk penggerjaan praktikum pada hari **Minggu, 15 Desember 2024**. Penyelesaian praktikum setelah tanggal tersebut (sebelum tanggal 19) **tidak akan mengurangi nilai kalian** sama sekali, namun bagi yang menyelesaikan **seluruh soal wajib sebelum soft deadline** akan mendapatkan **nilai bonus** yang dapat digunakan untuk mengurangi kekurangan nilai di praktikum lainnya.

IV. Langkah Umum Pengerjaan

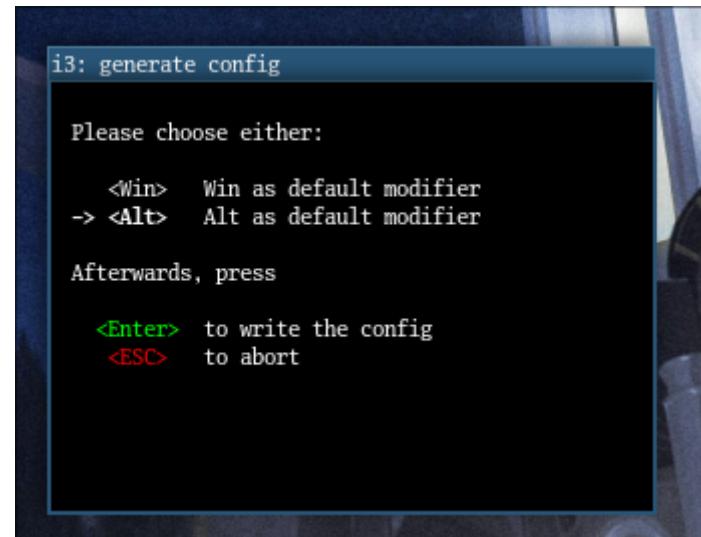
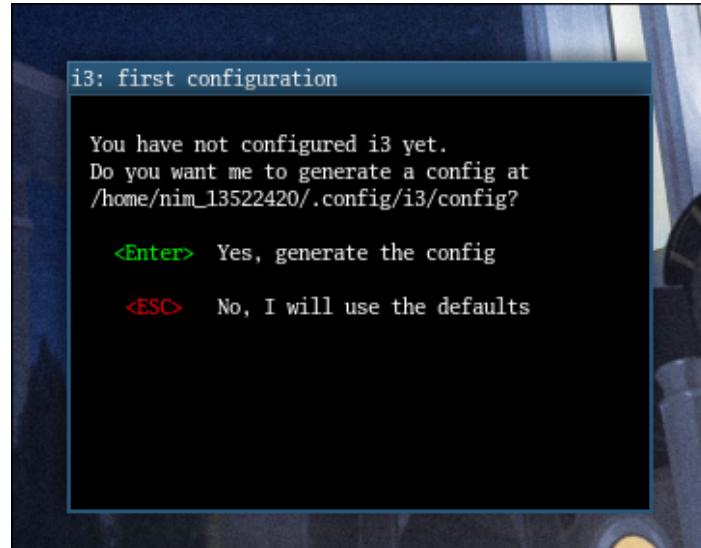
Karena praktikum menggunakan sistem operasi yang sama dengan praktikum sebelumnya, langkah-langkah awal pengerjaan akan sama seperti pada praktikum sebelumnya. Namun perlu diperhatikan bahwa terdapat **beberapa perubahan pada file .iso**, sehingga perlu dilakukan **pengunduhan ulang** menggunakan *link* yang tersedia pada dokumen panduan.

1. Ikuti panduan *setup* dan *run* file .iso yang tersedia pada pranala berikut:
 [Panduan Setup Praktikum 0x1 IF1230-II2130](#)
2. Setelah file .iso berhasil dijalankan, praktikan akan diminta untuk memasukkan NIM dan password yang telah dikirim ke milis praktikan masing-masing. Pastikan Anda terkoneksi ke internet pada langkah ini, karena proses autentikasi akan dilakukan di server.



Gambar 1. Menu login

3. Setelah autentikasi berhasil, praktikan akan masuk ke menu *interface* sistem operasi, dan pertama akan diminta untuk mengkonfigurasikan **mod key**.



Gambar 2. Menu konfigurasi mod key

Cukup tekan **Enter** untuk melakukan generasi config, lalu pilih **Alt** atau **Windows** sebagai mod key. Pemilihan mod key akan menentukan *shortcut* untuk membuka dan menutup terminal.

4. Praktikan lalu dapat mengakses *interface* dari sistem operasi.



Gambar 3. Wallpaper iso

Pengerjaan praktikum akan sepenuhnya dilakukan di terminal, sehingga dua shortcut penting yang perlu diketahui hanyalah:

- **Mod + Enter:** Membuka terminal
- **Mod + Shift + Q:** Menutup terminal (opsional, namun dapat mencegah terjadinya hal seperti pada **Gambar 4**)

Dengan **Mod** berupa tombol **Win/Alt** yang dipilih pada konfigurasi sebelumnya.



Gambar 4. I need more terminals

5. Praktikan akan mencoba menyelesaikan *binary bomb* yang terdapat pada file **asmciv**. Terdapat dua soal bonus, yaitu file *binary* **Freedom_Dive** dan **Latent_Kingdom**.

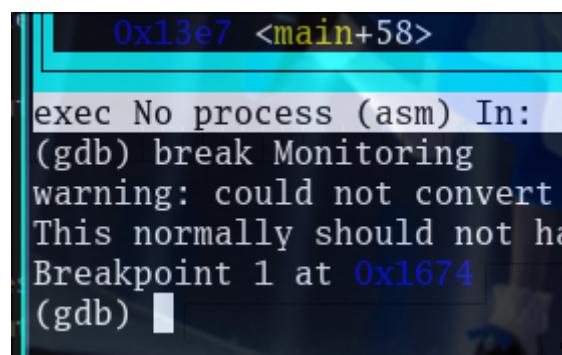
```
Machine View
Terminal
nim_13522420@box:~$ ls
Freedom_Dive  Latent_Kingdom  asmciv
nim_13522420@box:~$
```

Gambar 5. File-file *binary* praktikum

Tidak disarankan untuk langsung menjalankan file *binary*, silakan mengerjakan praktikum menggunakan *debugger GDB*.

6. [Opsiional] Disediakan varian GDB yaitu [peda](#), yang membuat fitur GDB lebih *user-friendly* dan mudah dipakai. Untuk menggunakan peda, cukup jalankan perintah `gdb_peda`.
7. Jalankan `gdb asmciv` untuk membuka file *binary* menggunakan GDB dan mulai mengerjakan soal.

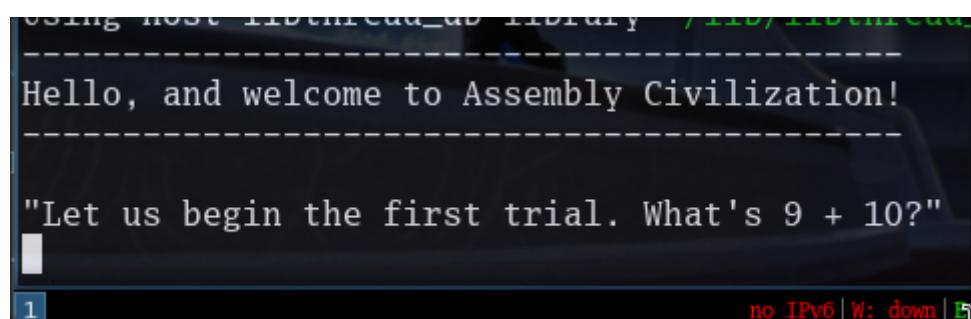
8. Terdapat panduan umum penggerjaan praktikum serta petunjuk singkat penggunaan GDB pada pranala berikut: [Panduan Praktikum 0x2 - Sister '22](#)
[Petunjuk Singkat GDB - Sister'22](#). Bagian selanjutnya hanya akan memaparkan instruksi umum penggerjaan soal.
9. Pada GDB, dapat dipasang *breakpoint* pada sebuah *line*, untuk melakukan *pause* eksekusi pada *line* tersebut. Pemasangan *breakpoint* dapat dilakukan dengan perintah **break [X]**, dengan X dapat berupa *address*, nama fungsi, atau sebuah *offset*. Informasi lebih lanjut tentang **break** dapat dilihat pada dokumen panduan.



The screenshot shows a terminal window for the GDB debugger. The title bar says "0x13e7 <main+58>". The main area displays the following text:
exec No process (asm) In:
(gdb) break Monitoring
warning: could not convert
This normally should not ha
Breakpoint 1 at 0x1674
(gdb)

Gambar 6. Pemasangan *breakpoint* pada fungsi Monitoring

10. Untuk mengeksekusi file *binary* pada GBD, dapat dilakukan dengan menjalankan command **run**.
11. Setelah file dijalankan, akan muncul *output* seperti berikut, dan praktikan akan diminta untuk memasukkan *input* jawaban.



The screenshot shows a terminal window with the following text:

Hello, and welcome to Assembly Civilization!

"Let us begin the first trial. What's 9 + 10?"
1 no IPv6 | W: down | E

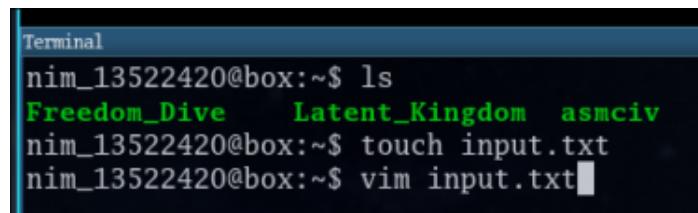
Gambar 7. *Output* program dan *prompt* untuk jawaban

Tugas praktikan pada praktikum ini adalah mencari *input* jawaban yang sesuai dengan melakukan *reverse engineering* terhadap kode *assembly* program.

12. Setiap menjalankan file *binary*, program akan selalu meminta input untuk keseluruhan soal, mulai dari soal pertama. Hal ini dapat menjadi pekerjaan yang meletihkan, terutama pada proses *solving* untuk soal-soal wajib terakhir.

Untuk mempermudah proses tersebut, program dapat dijalankan menggunakan sebuah file *input*, dengan langkah-langkah sebagai berikut:

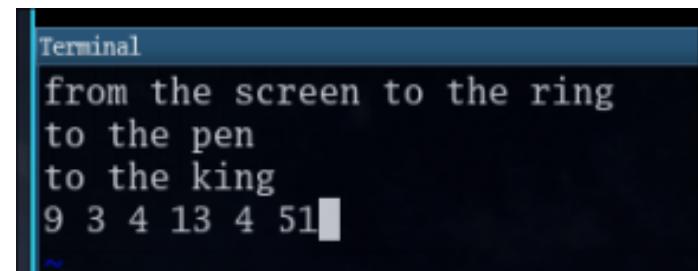
- Buatlah sebuah file baru pada *home directory*, misalkan 'input.txt'.



```
Terminal
nim_13522420@box:~$ ls
Freedom_Dive  Latent_Kingdom  asmciv
nim_13522420@box:~$ touch input.txt
nim_13522420@box:~$ vim input.txt
```

Gambar 8. Pembuatan file input.txt

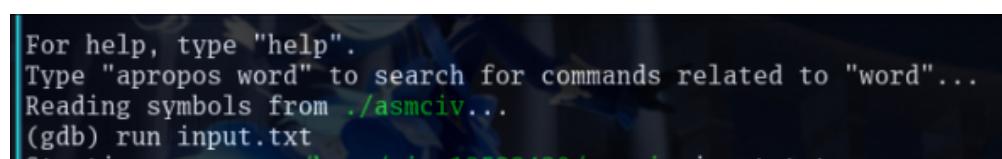
- Tuliskan jawaban untuk seluruh soal yang sudah *solved*, dimulai dengan soal pertama, terpisah dengan *newline* untuk masing-masing jawaban. Jangan lupa untuk menyimpan file.



```
Terminal
from the screen to the ring
to the pen
to the king
9 3 4 13 4 51
```

Gambar 9. Contoh penyimpanan jawaban pada file

- Ketika GDB sudah dijalankan dengan *file binary*, program dapat dijalankan menggunakan input dari input.txt menggunakan perintah berikut:



```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./asmciv...
(gdb) run input.txt
Starting program: /home/nim_13522420/asmciv/input.txt
```

Gambar 10. Menjalankan program menggunakan input file

PENTING: Karena eksekusi program selalu dimulai pada soal pertama, maka sebaiknya jawaban yang didapatkan selalu disimpan (baiknya di luar sistem operasi, agar tidak hilang ketika OS dimatikan). Jawaban yang benar akan langsung dikirim ke server, namun hal ini akan mencegah perlunya mencari ulang jawaban-jawaban sebelumnya jika terlupakan.

V. Pengumpulan dan Deliverables

1. Poin akan otomatis terkirim ke server setiap kali kalian memasukkan jawaban benar. Hal yang sama berlaku untuk *trap* dan pengurangan nilai.
2. Kumpulkan semua file C dengan format KodeC_[NIM].zip, pada situs (sisterlab.tech:8082). Format nama file untuk setiap soal adalah No[X]_[NIM].c (contoh: No1_13522022.c).

VI. Sistematika dan Peraturan Praktikum

1. Kalian dapat melihat nilai kalian yang tercatat pada server melalui halaman *profile* dan *scoreboard* di <http://sisterlab.tech:8082/>.
 2. **DILARANG KERAS** melakukan serangan Denial of Service (DoS) ataupun serangan lain terhadap server.
 3. **DILARANG** melakukan pengeluaran file *binary* dari sistem operasi atau melakukan *reverse engineering/exploitation* apapun (selain penggunaan GDB pada sistem operasi) untuk mengakali penggerjaan praktikum.
 4. **Waktu Mulai Praktikum** 3 Desember 2024 pukul 03:00 WIB waktu server.
Waktu Akhir Praktikum 19 Desember 2024 pukul 23:59 WIB waktu server.
 5. Pada tugas ini, Anda **diperbolehkan** menggunakan sumber-sumber eksternal, termasuk internet, *large language model* seperti ChatGPT, serta meminta bantuan teman. Namun, Anda sangat **dianjurkan** untuk mencoba mengerjakan sendiri terlebih dahulu. "*Percayalah, jika menemukan sendiri jawabannya, Anda akan mendapatkan sense of pride and accomplishment*" - Duke.
 6. **Mohon untuk tidak menanyakan sesuatu melalui surel**, gunakan QNA sebagai fasilitas untuk bertanya. Selalu utamakan untuk bertanya pada QNA agar orang yang mengalami masalah yang sama tidak perlu mengulangi pertanyaan yang ada.
 7. Terkecuali jika ada masalah pada server, **asisten akan menutup pelayanan QnA** (dan respons lainnya seperti DM) **6 jam sebelum deadline**. Artinya, *sudah bukan tanggung jawab kami apabila Anda mengalami masalah terkait tugas ini pada rentang waktu tersebut*.
- Kami mohon pengertiannya, sangat disarankan untuk tidak menunda penggerjaan tugas ini karena tugas ini dirancang dengan tingkat kompleksitas yang membutuhkan waktu dan perhatian yang cukup untuk diselesaikan dengan baik.
8. Anda dilarang *submit deliverables* dengan NIM orang lain. Tentunya Anda juga sangat dilarang melakukan plagiarisme dalam menulis file kode C. Tolong bertanggung jawab atas pekerjaan Anda sendiri.
 9. Kami akan menindaklanjuti segala bentuk kecurangan yang terstruktur, masif, dan sistematis.

10. Jika ada pertanyaan atau masalah penggerjaan (atau ada yang tidak sengaja melakukan submisi diluar format yang diberikan) harap segera mengirimkan pertanyaan ke **sheets QnA**: <https://bit.ly/QNAOrkomArsikom2024>
11. **SANGAT TIDAK DISARANKAN UNTUK DEADLINER DALAM PENGERJAAN PRAKTIKUM INI. YOU HAVE BEEN WARNED.**

VII. Referensi

1. **VirtualBox Snapshots** -
<https://howtogeek.com/150258/how-to-save-time-by-using-snapshots-in-virtualbox>
2. **GDB** -
https://docs.google.com/document/d/14DuYI91SBmym-K1aD7KTOVSxEUb_SI_S79qcFxEni8/edit?usp=sharing
3. **x86 Assembly** - <https://cdrdv2.intel.com/v1/dl/getContent/671110>
4. **I3wm** - <https://i3wm.org/docs/userguide.html>

VIII. Daftar Klarifikasi dan Revisi Soal

1. Sebelum mengerjakan nomor 7, `Last_Samurai`, Anda perlu **menjalankan perintah `set follow-fork-mode parent`** terlebih dahulu (dalam GDB). Pada fungsi nomor tersebut, terdapat sebuah potongan kode C yang melakukan *clear terminal*. Ini mengakibatkan eksekusi GDB menge-bug dan menghilangkan semua *breakpoint* seketika. Agar lebih aman, **pastikan Anda sudah menjalankan perintah tersebut sebelum menyelesaikan nomor 6.**

Asisten memohon maaf. Silakan mengontak kami melalui QnA atau japri jika terkena *trap* akibat hal ini.

Pesan dari Asisten



- Pol

<< No komen wkwkwk >>

- Owen

<< Yes komen >>

- Albert



<< awas kena trap(), pilih dengan tepat >>

- インドラ

<< mmf kalo pada ngomongin trap() semua ya ;;>>

- Flora



<< >>

« Jangan jadi trap, jadilah seperti larry »
- Onta



<< >>

« Ngerjain soal bonus biled, btw yang bikin soal bonus itu aku ges, kalo ada complain tentang soal bonus ke aku aja hehe~ »
- Layla



<< >>

- Rafiki



« 서둘러 오진 마 »

« 이 순간이 좋아 난 »

- Ryle



« trap() is love, trap() is life »

- Edbert



« gue kayaknya gapapa nikah ama trap() »

- Aldy