Lab 1
Stack and Stack Frame

Due: Thursday, September 24, 2026, 11:59 PM Eastern Time
Individual work | 5 points | Submit one PDF through D2L

Work window: September 11-24. These WolfCTF instructions and the September 24 deadline replace the Lab 1 connection instructions and dates in earlier handouts or slides.

1. Open your WolfCTF terminal

Sign in at https://portal.wolfctf.com/login using your WolfCTF **username** and password. Open **Labs > CSC 472 Fall 2026 > Stack and Stack Frame**, then click **Start Terminal**. Run the commands below in that browser terminal.

Direct lab: https://portal.wolfctf.com/labs/lab1
Account/terminal help: https://wcu.ninja/csc472/wolfctf

2. Copy and compile

```
mkdir -p ~/lab1-work
cp -n ~/courses/lab1/lab1.c ~/lab1-work/
cd ~/lab1-work
gcc -m32 -g -O0 -fno-omit-frame-pointer \
    -fno-pie -no-pie -fno-stack-protector lab1.c -o lab1
file lab1
./lab1
```

Work on your own copy in `~/lab1-work/`. The supplied program initializes p and q, calls `multiply_by_two`, and prints a result. Source download: lab1.c.

Check: `file lab1` must report a 32-bit Intel 80386 ELF executable, and the program must run. The flags select IA-32, debug information, no optimization, a frame pointer, a non-PIE executable, and no stack-canary setup for this teaching exercise. Stack addresses can still change between runs.

3. Start GDB + GEF (required)

```
gdb -q ./lab1
gef help
gef config context.layout 'regs code stack'
set disassembly-flavor intel
disassemble main
disassemble multiply_by_two
```

GEF loads automatically. Enter commands after the first line inside GDB. Confirm `gef help` works; do not use `-nx`. Record gcc/GDB versions, GEF startup information, build command, and architecture. Contact Dr. Chen if GEF fails to load.

4. Answer all five questions

Q1 - Constructing the frame

Identify the instructions that construct `main`'s stack frame. Give their addresses and explain the effect on ESP and EBP. Include any stack-alignment or saved-register instructions that occur in your build.

Q2 - Locating p and q

Find the instructions that initialize p and q. Explain the values written, operand size, and each variable's location relative to EBP.

Q3 - Explaining two copies of 3 and 4

Stop immediately before the call to `multiply_by_two`. Explain why copies of the values 3 and 4 appear in more than one place. Distinguish caller locals from the outgoing arguments using addresses and the instructions that placed them there.

Q4 - Reading the arithmetic

Explain what `add eax, edx` and `add eax, eax` do in `multiply_by_two`, and how they implement the C expression. Give a reasoned explanation for the compiler's choice instead of `mul`/`imul`.

Q5 - Proving the return value

Identify the register that carries the integer return value. Stop inside `multiply_by_two`, use `finish` to return to the caller, and verify the result before later instructions can change it.

Capture the two stack snapshots

Find the `call multiply_by_two` address in your own disassembly. Replace the placeholder below; do not copy an address from a classmate or a slide.

Caller
Immediately before the call

```
break *0xYOUR_CALL_ADDRESS
run
context
x/i $pc
info registers esp ebp
x/16xw $esp
```

Callee
After frame setup

```
break multiply_by_two
continue
context
x/i $pc
info frame
x/16xw $esp
bt
finish
info registers
```

Verify each stopping point; use `si` if needed to pass the callee's prologue. Include readable GEF `context` captures (registers, code, stack) at both stops. Keep snapshots separate and draw labeled caller/callee diagrams with addresses, ESP/EBP, locals, arguments, saved-frame/return-address data.

5. Prepare one PDF report

Filename: `Lastname_Firstname_Lab1.pdf`. Write in English. Use a separate title page, approximately 2-4 pages of main content, and an optional appendix. Use 11-12 pt body text, approximately 1-inch margins, page numbers, readable monospace code/output, and numbered figures with captions.

• **Title page:** lab title, CSC 472 / Fall 2026, your name, student ID, WolfCTF username, instructor, date performed, and submission date.

• **Introduction:** your objective and why stack-frame inspection matters, in your own words.

• **Environment and method:** GDB + GEF, architecture, compile command, Intel syntax, and where you paused. Include both GEF context captures.

• **Analysis and results:** Q1-Q5, each with a direct answer, your own evidence, and an interpretation. Include a labeled caller/callee stack diagram with real addresses.

• **Discussion/conclusion:** findings, a difficulty or unexpected result, and how you checked it.

• **References/assistance:** cite outside sources and AI assistance actually used; verify explanations using commands from your own session.

Read the detailed report guide: https://wcu.ninja/csc472/lab-report
Downloadable outline: CSC472_Lab1_Report_Outline.txt

Report rubric: 5 points total

Category	Weight	Points
Technical content (Q1-Q5 equally weighted)	50%	2.50
Organization	15%	0.75
Presentation	15%	0.75
Layout and visuals	20%	1.00

6. Submit through D2L

Open WCU D2L > your **CSC 472 Fall 2026** course > **Assignments > Lab 1**. Use **Add a File**, select your final PDF, finish the upload, and click **Submit**. Confirm the recorded filename and submission time; keep your PDF and confirmation.

Due: Thursday, September 24, 2026, 11:59 PM Eastern Time. Working in WolfCTF is not a D2L submission. If the folder is missing, closed, rejects PDFs, or shows a different date, contact Dr. Chen before the deadline.

Individual work: your explanations, captures, and diagrams must be your own. Follow the syllabus for academic integrity, AI use, and late work. D2L timestamps are authoritative. Never put passwords or session tokens in the report.