1. Common Debugging

1.1 Assembly-Specific Issues
Register Corruption: Accidental overwriting of caller-saved registers

Memory Access Violations:

bash
```
Segmentation fault (core dumped) # Common symptom
```
Stack Misalignment: ESP/RSP not 16-byte aligned before C calls

ABI Violations: Incorrect parameter passing (registers vs memory)

1.2 C/Assembly Interface Issues
Type mismatches between C prototypes and assembly code

Incorrect function calling conventions

Memory ownership conflicts (who frees allocated memory?)

2. Essential Tools
   2.1 GDB Configuration
   bash

Enhanced debugging setup

```
gdb -q ./build/asm_test
-ex "set disassembly-flavor intel"
-ex "layout asm"
-ex "break main"
```
2.2 Debug Build Flags
Makefile Additions:

makefile
```
CFLAGS += -ggdb3 -O0
ASMFLAGS += -F dwarf -g
```
3. Debugging Workflow
3.1 Basic Checks
Verify register usage with reg_tracker.py:

bash
```
python3 tools/reg_tracker.py build/asm_test
```
Check stack alignment at function boundaries

Validate memory accesses with mem_access.py:

python

Generate access pattern from log

python3 tools/mem_access.py memory.log

3.2 Assembly Function Tracing

gdb

(gdb) break sumOfDigits # Set breakpoint

(gdb) display /i $pc # Show next instruction

(gdb) info registers # View register states

(gdb) stepi # Single-step assembly

3.3 Mixed C/Assembly Debugging

gdb

(gdb) break c_wrapper.c:15 # C breakpoint

(gdb) next # Step over C code

(gdb) step # Enter assembly function

(gdb) finish # Return to C context

4. Advanced Techniques

4.1 Memory Inspection

View Stack Frame:

gdb

(gdb) x/16xw $rsp # Examine 16 words at stack pointer

(gdb) x/s $rdi # View string parameter

Heap Allocation Tracking:

bash

valgrind --track-origins=yes ./build/asm_test

4.2 Signal Handling

gdb

(gdb) handle SIGSEGV nostop noprint # Continue on segfault

(gdb) catch syscall # Trap system calls

4.3 Post-Mortem Analysis

bash

Generate core dump

ulimit -c unlimited

./build/asm_test

gdb ./build/asm_test core

5. Tool-Specific Guides

5.1 reg_tracker.py

Key Features:

Track register changes across instructions

Identify unexpected register modifications

Usage:

bash
```
python3 tools/reg_tracker.py -f reverseArray build/asm_test
```
5.2 mem_access.py
Pattern Recognition:

Sequential access: Linear memory regions (arrays)

Random access: Hash tables/pointer structures

Hot spots: Repeated access to same address

5.3 debug_helpers.py
Disassembly Inspection:

bash
```
python3 tools/debug_helpers.py -d build/assembly_library.o
```
6. Troubleshooting Checklist
Symptom First Checks
Segmentation Fault 1. Stack pointer alignment
7. NULL pointer dereference
Incorrect Calculations 1. Register width mismatches
8. Overflow handling
Infinite Loop 1. Missing loop termination
9. EFLAGS corruption
GDB "No Symbol" Errors 1. Debug symbols in Makefile
10. Stripped binaries
11. Performance Debugging
7.1 Cycle Counting
gdb
```
(gdb) perf record ./build/asm_test
(gdb) perf annotate -M intel
```
7.2 Cache Analysis
bash
```
valgrind --tool=cachegrind ./build/asm_test
cg_annotate cachegrind.out.
```
12. FAQ
Q: How to debug SSE/AVX registers?
A: Use:

gdb
```
(gdb) info all-registers
(gdb) p $ymm0
```
Q: Why does printf crash in assembly?
A: Check:

Stack alignment (must be 16-byte before call)

XMM register preservation

Q: How to debug ABI issues?

A: Use GDB's calling convention checker:

```
gdb
(gdb) set check-abi on
```