Homework 1 Mistakes.

→ no need to use inline asm.

→ Mistakes related to pointer variables in C.

Array \[
\text{value}[0] \\
\vdots \\
\text{value}[n-1]
\]

\[\text{Memory allocated for set of values in an array}\]

For a pointer we need to assign memory before it can be used.
char line[N] [global variable]

fgets(line...

If instead of being a global variable 'line' is a local variable

get args

line [80]

3

$q2()$

getargs()
Local variables have a life time of function.

→ exit(1) — To solve seg fault for q3 & q4.
Save system stack using do-switch ( &sys_stack, spt ) and then call exit() in bvc which eg calls do-switch ( sys_stack )

Q4: getline ( buf )
   P1: i = get_input ( buf )
   if i != current
      do-switch

return

In above case but will goto the wrong process.
- Instead use a global variable
  & assign buf with global after switching.
  
  i = get_input (global)
  if i ≠ current
     do_switch
     buf ← global

---

**More Virtual Memory**

*page size*

VMS had a page size of 512 bytes.

Variable length instruction
- Instruction could span across two pages

Instr{[ | ]} → [ | ] [ | ] [ | ]
Minimum no. of pages required in physical memory to execute an instruction was > 64

User program: 8

PFHandler [worst case not another 8 pages] > 64 pages.

For a total of 128 K memory 512 byte page size was then required in above scenario.
Shared Memory

Useful for shared libraries and dll's.

Typically, a bulk of a program is the library.
Eg for `/bin/ls 100K 2M
xterm 800K 4M

Make shared library look like a separate executable and have them mapped to virtual Address of a program.

```
/lib/libc

/bin/ls
```

- Problem: Where are fn located in shared library.
- Soln: Do Final stage of linking in 2 different place.

1) load time: patch list
2) indirect table.
2) runtime: put a stub in indirection table
   ie instead of printf replace by
   load ('printf') ← search for printf
   to resolve reference to 'printf'.

Page Replacement - VMS, Linux

VMS - Did not have access bit in
      Page tables. Instead did fall.

FIFO

\[
\begin{array}{c}
\text{LRU List} \\
\text{mapped pages}
\end{array}
\]

\[\text{pf → VA → file, offset}\]

\[\text{Maps to a} \]
\[\text{waage in LRU list.}\]
Linux

Clock

mapped
pages

LRU
list

→ VMS does per process allocation i.e. evict a page from faulting process

→ Linux - Global : Clock : PNBIS all of memory, if a page needs to be evicted it is taken from tail of LRU list.
Working Set:

\[ ws(\tau) \] - pages accessed in period
\[ \tau \leq \text{now} - (\text{now} - \tau) \]

"Foot print": \( ws(\infty) \)

Working set grows with the time.
Constant \( \tau \). We can also look at \( \tau \) using "Miss Ratio Curve" (MRC)

MRC:

\[ \text{Miss rate} \]
\[ \text{Memory Size} \]

Show that Linux - Global Scheme is better for overall performance.
Walkthrough

OS Virtual Addr space

FFF000xxx - Stack alloc zero filled

09000xxx - data alloc zero

(2 Instr)

MOV (09000000), EAX

PUSH EAX

I

08000xxx - code /bin/xyz

Initial

SP = FFF01000

PC = 08000000

Assum page directory is allocated.

CR3 → PDB → EB → DB → CB → AB → XYZ
1) Attempt to execute 08000000.
2) Fault [address not mapped]
3) Allocate block B.
4) Read xyz → block
5) Allocate page table A
6) Set up PTEs, return.
7) Attempt to execute 08000000 again.
8) Fault on 09000700
9) Allocate Zero page C
10) Setup PT D
11) Setup D → C
12) Resume
14) Attempt to execute 08000000
15) Success
16) Attempt to execute 08000004
17) Similarly as above allocate F & E.