DispatchMessage(&msg) , MSG | struct tagMSG

msg.hwnd	→ HWND
msg.uMsg	→ UINT
msg.wParam	→ Information-1
msg.lParam	→ Information-2
msg.time	→ timestamp
msg.pt	→ Point

.x
.y

1) Handle message
2) Handle event
3) Dispatch message
4) Dispatch event

} to call a call-back procedure registered by an application in the window class of window.

hwnd → Handle आपल्या window →
CreateWindowEx → 2nd parameter
→ Window class name → WNDCLASSEX
→ wnd.lpfnWndProc → Window च्या Window procedure चा Addr. असतो.

1) Handle messag
2) Handle event
3) Dispatch messa

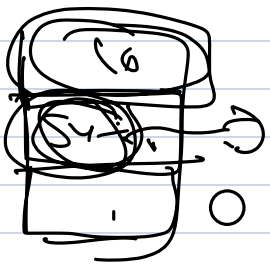
Inside O.S.

4) Dispatch event

```
VOID DispatchMessage(PMSG pMsg)
{
    PWNDCLASSEX pWnd;
    pWnd = AbrakaDabra(pMsg->hwnd)
    [ pWnd->lpfnWndProc( pMsg->hwnd, pMsg->uMsg,
                         pMsg->wParam, pMsg->lParam;
    );]
}

RET SysCall(PARAM1, --, PARAMIC)
{
    Detect which process/thread called
    this SysCall → AbrakaDabra
    {
        [System call is always
        executed in the context
        of the calling process.
    }
    → Chapter-6 : SysHandler | context
}
```

(11)

[The kernel always executes in the current context layer of process.]



Prof. Assembly Richard <0, 5, 6, 7, 11

Advanced 80386 Prog. : 1-> Skim. 2, 3, 4, 5, 6 + 10 times

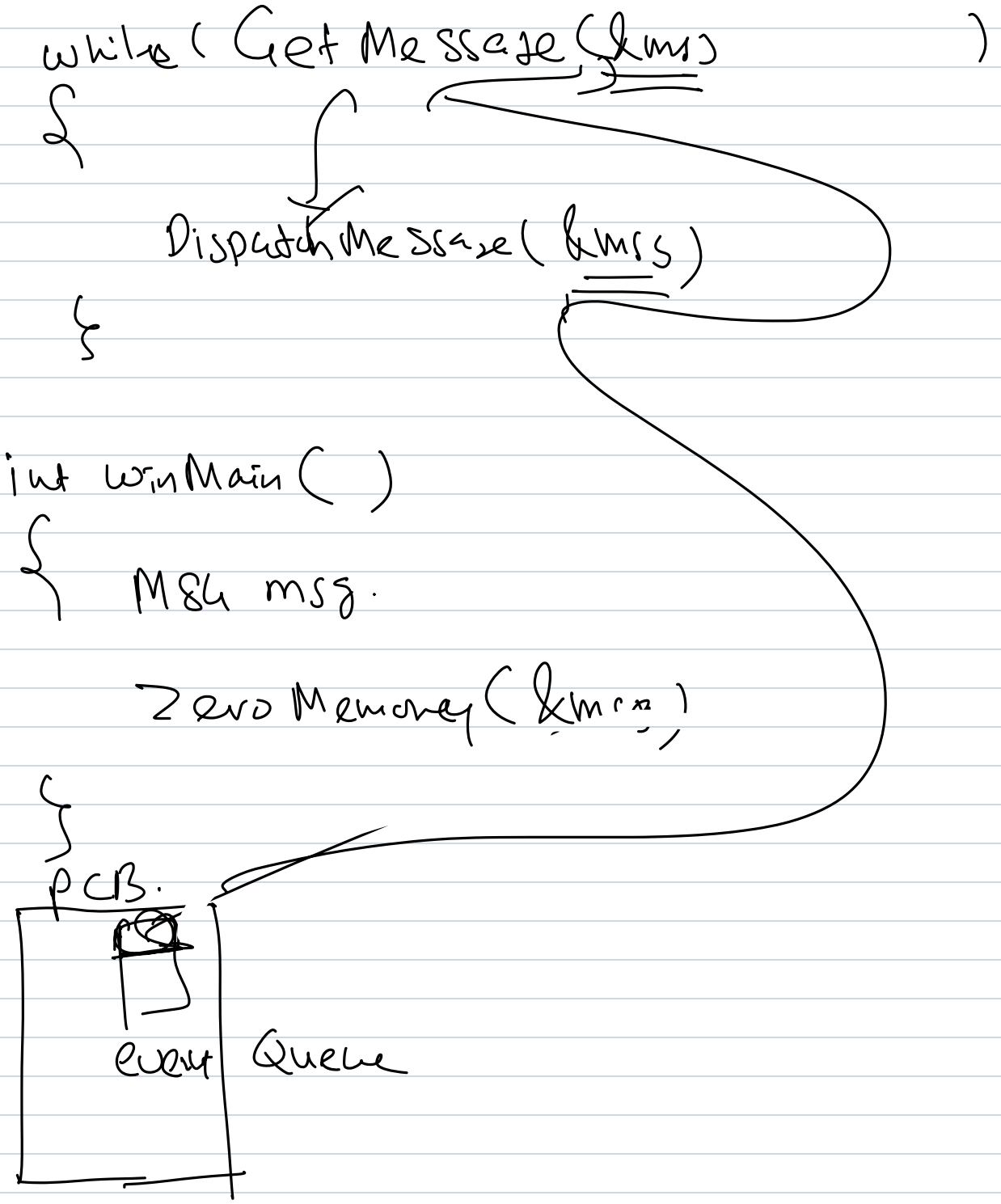
Linux compilation:

LSP: VFS,

LKMPG: basic char driver.

UTLK: chapter 5.

```
LRESULT CALLBACK WndProc(HWND hwnd,
                         UINT uMsg,
                         WPARAM wParam,
                         LPARAM lParam
{

}

{
                    An
DispatchMessage(PMSG pMSG)
{
    PWNDCLASSEX pWnd;
    pWnd = AbraKaDabra(pMsg->hwnd)
    pWnd->lpfnWndProc(pMsg->hwnd, pMsg->uMsg,
                      pMsg->wParam, pMsg->lParam)
}
```

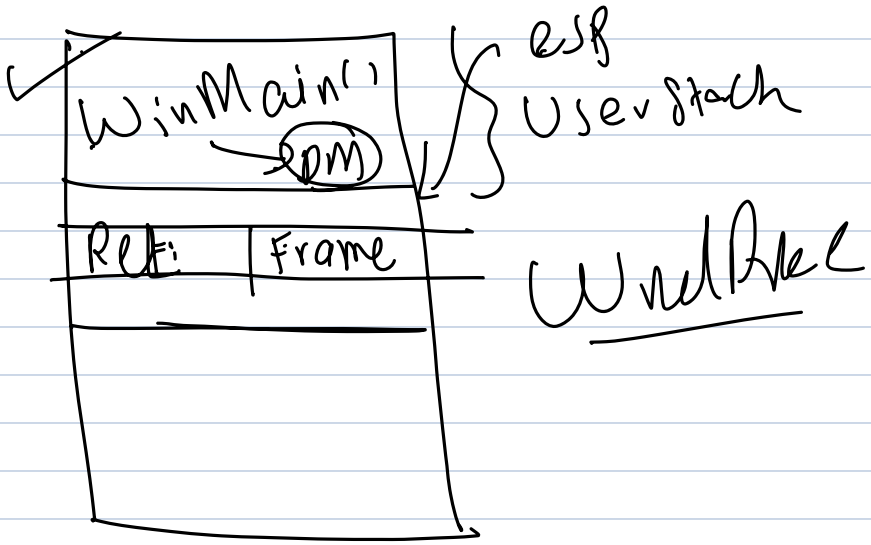


```
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    if(uMsg == EVENT_ID)
    {
        // HANDLER

        return (0);
    }
    else if(uMsg== EVENT_ID)
```

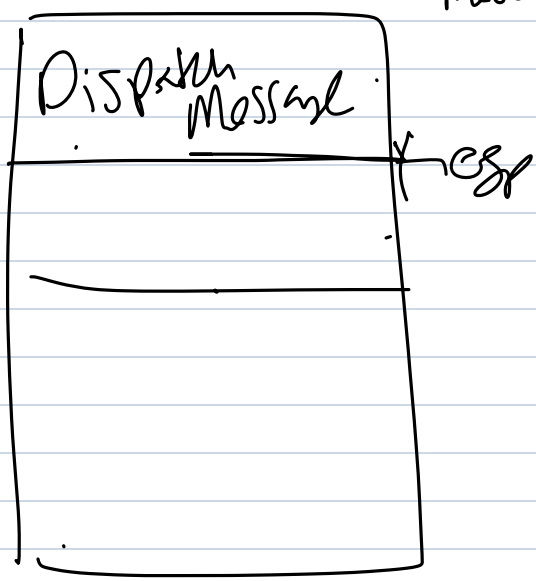
```
    {
        // HANDLER

        return (0);
    }


    return DefWindowProc(hwnd, uMsg, wParam, lParam);

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    if(uMsg==WM_RBUTTONDOWN)
    {
        MessageBox(NULL, TEXT("Right Button Was Clicked"), TEXT("WndProc"), MB_TOPMOST | MB_OK);
        return (0);
    }
    else if(uMsg == WM_DESTROY)
    {
        PostQuitMessage(0);
        return (0);
    }

}

Int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
    szClassName;
    szWindowTitle;
    WNDCLASSEX wnd;
    MSG msg;
    HWND hwnd;

    wnd.this = something;
    wnd.that = something;

    RegisterClassEx(&wnd);

    hwnd = CreateWindowEx(params);

    ShowWindow(hwnd, nShowCmd);
    UpdateWindow(hWnd);
```

```
    while(GetMessage(&msg, NULL, 0, 0)
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg); // pWnd = AbraKaBadra(pMsg->hwnd),
                               // pWnd->lpfnWndProc();
    }


}
```

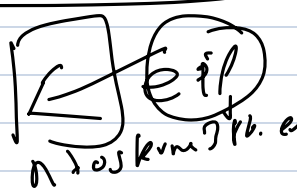
At the time of calling DispatchMessage Application -> User Stack

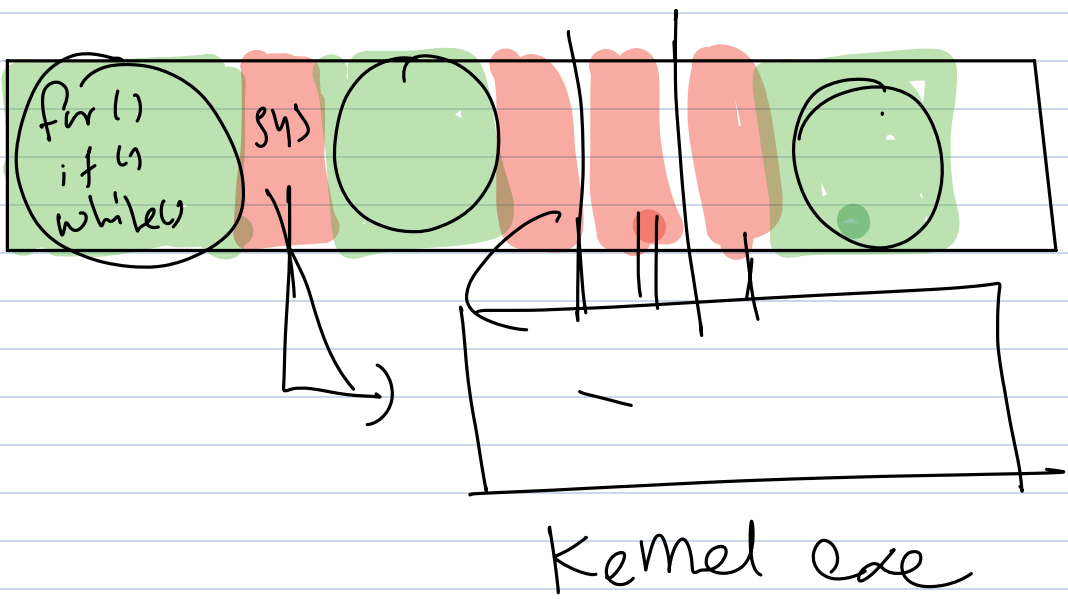


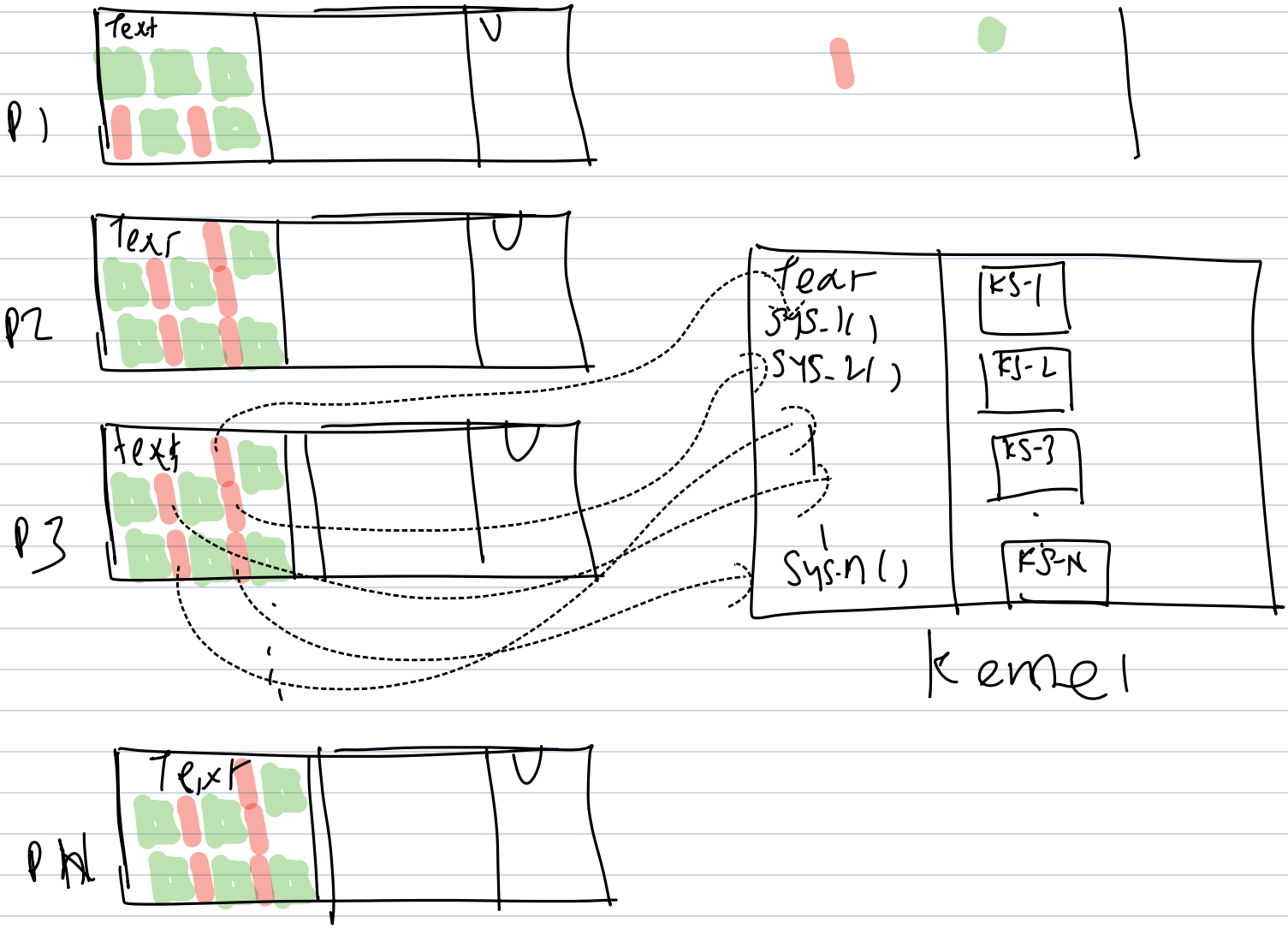
After calling DispatchMessage



app → kernel call
kernel → app return





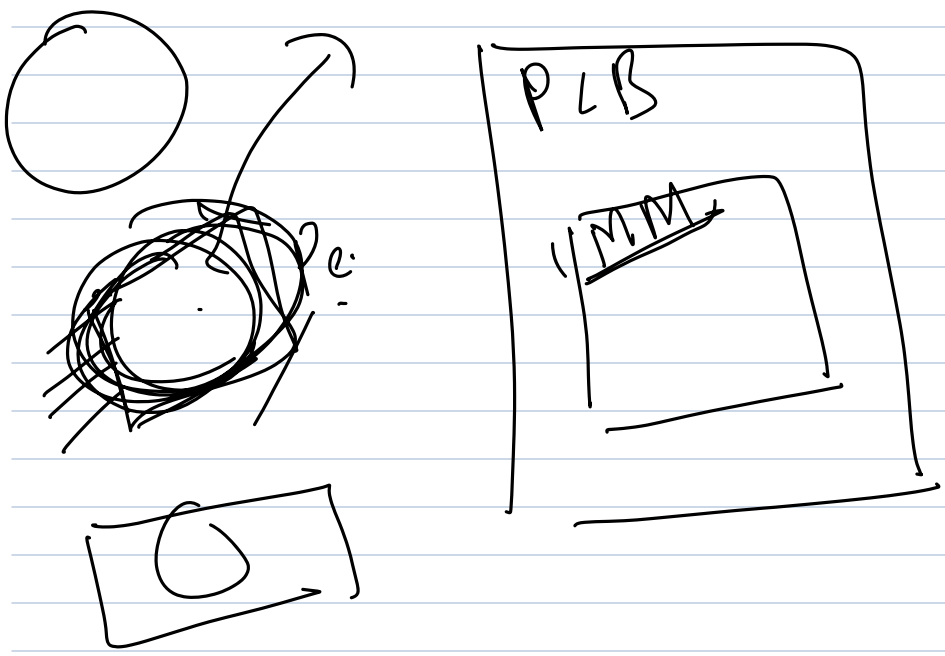


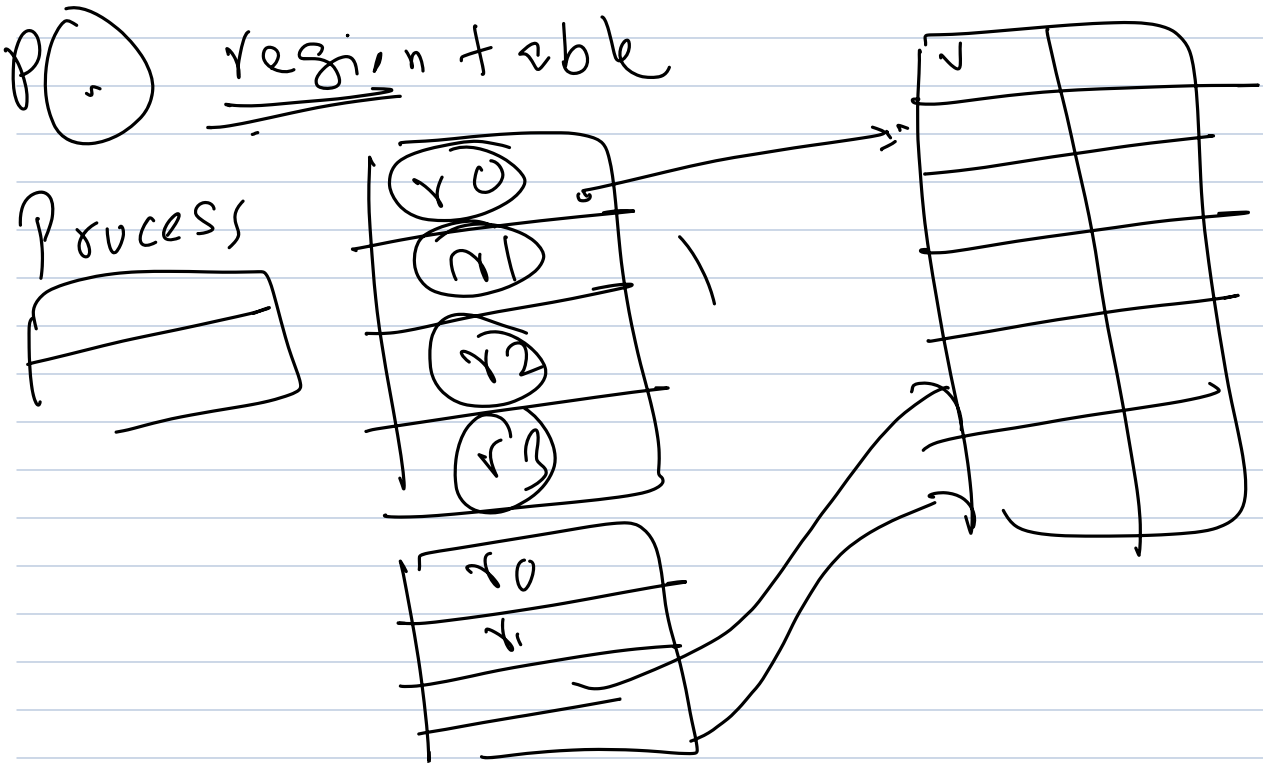
Movl %esp, %ecx

Addl $0xfffff000, %ecx

Movl (%ecx), %eax

struct task.




PCB
region table
Process
r0
r1
r2
r3
r0
r1
V