ব্যবহারকারীর পাতা

আহসান হাবীব পলাশ

TSR প্রোগ্রামিং

অতন্দ্র প্রহরী

'কমপিউটার ভাইরাস' জিনিসটি কমপিউটার প্রেমীদের কাছে কম বেশী পরিচিত। আজ রাতে কোন জরুরী কাজ করা হলো। কাল দরকার। সকালে দেখা গেল সব শেষ। ডিস্কে সব ফাইল আছে। গতকালেরটা নাই। কাজটি কার বুঝতে অসুবিধা হয় না। তখনকার মানসিক অবস্থা শুধুমাত্র ভুক্তভোগীরাই বুঝতে পারে। এই কারণে সাবধানীরা ভাইরাস গার্ড (Virus Guard) ব্যবহার করে। যেটা ভাইরাস সমৃদ্ধ ডিস্ক কমপিউটারে প্রবেশ করানো মাত্র সাবধান করে দেয়।

কাজটি করার জন্য ভ্গার্ডস্তুকে সর্বদা সতর্ক থাকতে হয়, অতন্দ্র প্রহরীর মতো। এটা এমন একটা প্রোগ্রাম যা রান করলে মেমোরীতে থেকে যায় এবং দরকার মতো প্রয়োজনীয় রুটিনগুলো এক্সিকিউট করে। TSR প্রোগ্রামের একটা চমৎকার উদাহরণ গার্ড। আরো অনেক TSR প্রোগ্রাম আমরা নিত্য ব্যবহার করে থাকি। যেমন doskey.com।

আনুষঙ্গিক

TSR প্রোগ্রাম লেখার জন্য কিছু জিনিস জানা আবশ্যক। একটা কমপিউটারকে যখন Boot করা হয় তখন কিছু রুটিন মেমোরীতে লোড হয়। রুটিনগুলোকে বলা হয় Interrupt Service Routine (ISR)। এই রুটিনগুলো কিছু DOS (Disk Operating System) এবং কিছু BIOS (Basic Input Output System) এর। রুটিনগুলোর অ্যাড্রেস RAM (Random Access Memory) এর প্রথমে টেবিল আকারে সাজানো থাকে। একে বলে ভেক্টর টেবিল। টেবিলে ২৫৬টি অ্যাড্রেস পরপর সাজানো থাকে। প্রত্যেকটির জন্য চার বাইটস্ করে বরাদ্দ (১ বাইট=৮ বিট)। এই রুটিনগুলো কল করা হয় নম্বরের মাধ্যমে। নম্বরগুলো হলো ০–২৫৫।এ গুলোকে ইন্টারাপ্ট নম্বরও বলা হয়। ইন্টারাপ্ট হলো CPU-এর কাছে প্রেরিত একটা সংকেত, যার অর্থ একটা রুটিন এক্সিকিউট করা খুব জরুরী।CPU তখন সম্ভব হলে আনুষঙ্গিক কাজ সেরে যে ইন্টারাপ্ট পাঠিয়েছে তার কাছ থেকে একটা নম্বর (Interrupt Number) গ্রহণ করে। নম্বর অনুসারে ভেক্টর টেবিল থেকে রুটিনটার অ্যাড্রেস বের করে। তারপর ঐ রুটিনকে এক্সিকিউট করে পূর্বের কাজে ফিরে যায়।

কোন প্রোগ্রামার যেমন প্রোগ্রামের মাধ্যমে ইন্টারাপ্ট করতে পারে তেমনি CPU নিজ প্রয়োজনে বিভিন্ন সময় ইন্টারাপ্ট করে থাকে। যেমন কোন সময় কোন ক্যালকুলেশনে ভাজক শূন্য হয়ে গেলে 'Divide by Zero' মেসেজটি স্ক্রীণে দেখা যায়। এটা একটা ইন্টার্নাল ইন্টারাপ্টের উদাহরণ।

কোন কোন ইন্টারাপ্টের আবার একাধিক সার্ভিস রুটিন আছে। যেমন 0 x 10 (Hex 10) একটা BIOS ভিডিও ইন্টারাপ্ট। এর অনেকগুলো সার্ভিস রুটিন আছে। যার এক নম্বর সার্ভিস রুটিনের মাধ্যমে আমরা কার্সর কে বিভিন্ন সাইজে পরিবর্তন করতে পারি। ব্যাপারটা অনেকটা কোন মেনুর সাব মেনুর মতো। যে সমস্ত ইন্টারাপ্ট আমাদের কাজে লাগবে সেগুলো নিয়ে এখানে আলোচনা করা হবে। অন্যান্যগুলোর সম্পর্কে জানতে হলে ডস-এর কোন ভাল বইয়ের সাহায্য নেওয়া যেতে পারে। একটা কথা বলে রাখা ভালো, ভেক্টর টেবিলের পুরোটায় এখনও ISR এর অ্যাড্রেস দ্বারা পূর্ণ হয়নি। কিছু কিছু অংশ এখনও খালি আছে যেমন 0 X 64।

TSR

কোন প্রোগ্রাম যখন রান করা হয় তখন ডস র‍্যাম এ পুরো প্রোগ্রামটা লোড করে। তবে অনেক বড় বড় প্রোগ্রামের ক্ষেত্রে যখন যে অংশটুকু দরকার শুধুমাত্র সেটুকু লোড হয়। প্রোগ্রামটা রান করা শেষে যখন নতুন কোন প্রোগ্রাম রান করা হয় তখন পূর্বের অংশেই লোড হতে পারে। অর্থাৎ রান করার পর ডস-এর কাছে প্রোগ্রামটার কোন মূল্য থাকে না। কিন্তু কোন TSR প্রোগ্রাম রান করালে ডস একটা অংশ TSR প্রোগ্রামের জন্য রিজার্ভ রাখে। পরে নতুন কোন প্রোগ্রাম রান করালে রিজার্ভ অংশটুকু বাদ দিয়ে তারপর থেকে লোড করে। এই জন্য আসলে TSR প্রোগ্রাম বলা হয়। Terminate and Stay Resident। Terminate করা সত্ত্বেও সে মেমোরীতে অবস্থান করে। ডস কোন টিএসআর প্রোগ্রামের জন্য কতটা অংশ রিজার্ভ রাখবে তা প্রোগ্রামারকেই বলে দিতে হবে। কাজটি করার জন্য Turbo C এর ফাংশন keep(ret-code, Size) ব্যবহার করা যেতে পারে।

এই সাইজ দেখে ডস বুঝতে পারে কতগুলো প্যারাগ্রাফ রিজার্ভ রাখতে হবে। প্রত্যেক প্যারাগ্রাফ হলো ১৬ বাইটসের। সুতরাং ২০ কিলোবাইটসের কোন প্রোগ্রামের জন্য সাইজ-এর মান ১৫০০ দিলেই যথেষ্ট। খুব উন্নতমানের প্রোগ্রাম লিখতে চাইলে, প্রোগ্রাম মেমোরীতে ঠিক কত জায়গা দখল করে তা বের করে সাইজ-এর মান সঠিকভাবে বের করা উচিত। কীপ ফাংশনের কাজ প্রোগ্রামকে রেসিডেন্ট বানানো। অর্থাৎ টিএসআর এর জন্য কীপ অপরিহার্য। Keep ret-code এর মান ডস-এর কাছে পাঠিয়ে দেয়। যারা Assembly ল্যাংগুয়েজ ভাল জানেন খুব সহজেই কীপ ফাংশনটি বানাতে পারবেন। 0x21 ইন্টারাপ্টের 0x49 সার্ভিস রুটিন কোন প্রোগ্রামকে TSR বানায়। সুতরাং কীপ ফাংশনটি হবে নিম্নরূপ।

```
Keep (unsigned ret–code, unsigned size)
{
union   REGS  r;
r.h.ah = 49 ; /* Terminate but stay Resident */
r.h.al=ret_code;
r.x.dx=size;
int86 (0x21, &r, &r);
}
```

একটা TSR প্রোগ্রামের দুটো অংশ থাকে। প্রথম অংশের কাজ হলো প্রোগ্রামটিকে initialize করা এবং ভেক্টর টেবিলে নতুন রুটিনের অ্যাড্রেস বসানো। ২য় অংশে TSR কি কাজ করবে তা বলা থাকে।

কোন TSR প্রোগ্রামকে কী-এর মাধ্যমে বাহিরে থেকে invoke করার ব্যবস্থা করা যায়। এই কী-কে hot key বলে। hot key প্রেস করা মাত্র TSR প্রোগ্রামের ২য় অংশটি তার নির্ধারিত কাজ করে।

ভেক্টর টেবিল থেকে কোন অ্যাড্রেস পড়া বা ভেক্টর টেবিলে নতুন কোন অ্যাড্রেস বসানোর জন্য TURBO C-এর নিম্নলিখিত ফাংশনগুলো ব্যবহার করা হয়।

getvect : এটা প্যারামিটারে দেয়া ইন্টারাপ্ট নম্বর অনুসারে ভেক্টর টেবিল থেকে ISR এর অ্যাড্রেস পড়ে। তারপর far pointer এর মাধ্যমে এর মান পাঠিয়ে দেয়।

উদাহরণ,

```
void  Interrupt (*p) (void); /*declaration */
P=getvect(5);
```

এখানে P একটা far pointer যা ইন্টারাপ্ট 5-এর ISR-কে পয়েন্ট করে আছে।
setvect : কোন নতুন ইন্টারাপ্ট ফাংশনের অ্যাড্রেস ভেক্টর টেবিলে বসাতে এটা ব্যবহৃত হয়।

```
Format : Void setvect (int intr_no, void interrupt (*ISR) ());
```

উদাহরণ ঃ

```
void interrupt  new_5 ( ); /*declaration */
setvect (5, new_5);
```

এখানে new-5 রুটিনের অ্যাড্রেস ভেক্টর টেবিলে বসানোর পর রুটিনটি ইন্টারাপ্ট 5 এর ISR হিসাবে কাজ করে। সুতরাং এটাকে অবশ্যই ইন্টারাপ্ট রুটিন হিসাবে ডিক্লেয়ার করতে হবে।

সহায়ক

আগেই বলা হয়েছে TSR প্রোগ্রাম রান করলে সেটা মেমোরীতে অবস্থান করে এবং প্রয়োজন মতো রুটিন এক্সিকিউট করে। প্রোগ্রাম রান করানোর পর তার উপরে ব্যবহারকারীর কোন নিয়ন্ত্রণ থাকে না। তাহলে দরকার মতো রুটিনকে এক্সিকিউট করাবে কে? এমন একটা ব্যবস্থা করতে হবে যাতে রুটিনটা এক্সিকিউট করার জন্য সর্বদা প্রস্তুত থাকে। বাহির (hot key) বা ভিতর থেকে কোন সংকেত পাওয়া মাত্র উপযুক্ত পরিবেশ থাকলেই যেন রুটিন এক্সিকিউট করতে পারে। এই সমস্যা সমাধানে timer ইন্টারাপ্টকে ব্যবহার করা হয়। প্রত্যেক system-এ একটা টাইমার চিপ আছে যেটা প্রতি সেকেণ্ডে 18.2 বার CPU কে ইন্টারাপ্ট করে। CPU তখন ইন্টারাপ্ট OX 8 এর ISR এক্সিকিউট করে। ইন্টারাপ্ট OXIC ও টাইমার সংক্রান্ত। তবে এখানে OX8 নিয়েই আলোচনা করা হবে। সুতরাং কোন TSR রুটিনের অ্যাড্রেস যদি ভেক্টর টেবিলে ইন্টারাপ্ট OX8 এর ISR এর অ্যাড্রেসে বসিয়ে দেওয়া যায় তবে প্রতিবার ইন্টারাপ্টের সাথে সাথে TSR রুটিনটা এক্সিকিউট করবে। রুটিনটা যেহেতু সেকেণ্ডে 18 বার কল করা হয় তাই মনে হবে সব সময়ই এটা এক্সিকিউট করতে প্রস্তুত। এখন hot key প্রেস করা হয়েছে কিনা চেক করে এক্সিকিউশন নিয়ন্ত্রণ করা যায়। অর্থাৎ ইচ্ছামতো শর্ত জুড়ে, নিজের পছন্দ মতো কাজ করে নেওয়া যায়। তবে ঐ অ্যাড্রেস যে অরিজিনাল রুটিনের তাকেও এক্সিকিউট করাতে হবে। তা না হলে সিস্টেমের কিছু কাজ অপূর্ণ থেকে যাবে। পরিণতিতে সিস্টেম ক্রাশ করতে পারে। ভেক্টর টেবিলে নতুন রুটিনের অ্যাড্রেস বসানোর পূর্বেই ঐ অ্যাড্রেসটি getvect-এর মাধ্যমে পড়ে সেভ করে রাখতে হবে।

নতুন যে ISR লেখা হবে তার মধ্যে সেভ করে রাখা অ্যাড্রেসের ISRটা এক্সিকিউট করালেই যথেষ্ট।

এখানে সাবধানতার কিছুটা অবকাশ রয়ে গেছে। নতুন ISR এক্সিকিউট করা মানে পূর্বের কাজের সাথে সাথে নতুন কিছু কাজ করে। কিন্তু ডস এক সাথে একটার বেশী কাজ করতে পারে না। সুতরাং দেখতে হবে কখন ইন্টারাপ্ট করা নিরাপদ। এটা বোঝার দুটো উপায় আছে।

(১) কোন ইনপুটের জন্য ডস-এর অপেক্ষা করা একটা লুপের মতো। এই লুপে ইন্টারাপ্ট OX28 এক্সিকিউট হয়। এই সময় ডস-কে ইন্টারাপ্ট করা নিরাপদ। আবার সুবিধা হলো ডস ইনপুটের জন্য অনেক সময় অপেক্ষা করলেও TSR এর নিজের কাজ করতে কোন অসুবিধা হয় না। ভেক্টর টেবিলে ইন্টারাপ্ট OX28-এর অ্যাড্রেস TSR রুটিনের অ্যাড্রেস বসিয়ে (OX8 এর মতো) এই সুবিধা পাওয়া যায়। তবে অনেক প্যাকেজ ডস-এর মাধ্যমে না করে BIOS এর মাধ্যমে I/O (Input/output) অপারেশন করে। সেক্ষেত্রে আমাদেরকে দ্বিতীয় উপায় দেখতে হবে।

(২) ডস যখন অ্যাকটিভ থাকে তখন একটা flag সেট (১) করে রাখে। আর যখন ইনঅ্যাকটিভ থাকে তখন flag টা রিসেট (০) করে দেয়। এই flag-এর লোকেশন OX34 DOS ফাংশনের মাধ্যমে পাওয়া যায়। প্রোগ্রামের শুরুতে একটা far pointer সেট করতে হবে যা DOS কি অবস্থায় আছে বলে দেবে। DOS যখন ইনঅ্যাকটিভ থাকবে তখন অনায়াসে TSR রুটিন এক্সিকিউট করতে পারবে।

এত কিছুর পরও একটা সমস্যা থেকে যায়। ডিস্ক I/O অপারেশনের সময়ে CPU অন্যের কাছে কণ্ট্রোল দিয়ে দেয়। ফলে timer ইন্টারাপ্ট যতই বিট দিক না কেন CPU এর কিছুই করার থাকে না। সুতরাং এই সময়টুকুতে TSR কে চুপচাপ বসে থাকতে হবে।

কাঠামো

কোন TSR প্রোগ্রামের জন্য যে কাজগুলো সাধারণতঃ করতে হয় সেগুলো নীচে দেওয়া হলো।

(১) কোন TSR প্রোগ্রামকে এক বারের বেশী মেমোরীতে লোড করা উচিত না। কারণ এটা সিস্টেমের মেমোরী ব্যবহার করে। সুতরাং প্রোগ্রামের শুরুতে চেক করতে হবে প্রোগ্রামটা আগেই Install করা হয়েছে কি না। এই কাজের জন্য ইন্টারাপ্ট OX64 কে ব্যবহার করা যেতে পারে। এটি একটি অব্যবহৃত ইন্টারাপ্ট, getvect () ফাংশনের সাহায্যে দেখতে হবে OX64-এর অ্যাড্রেস NULL কি না। NULL মানে সেখানে কিছুই নেই। NULL পেলে TSR কে install করা যাবে। তবে সাথে সাথে setvect () এর মাধ্যমে একটা flag সেট করে দিতে হবে যাতে পুনরায় Install করা না যায়।

```
old-int64=getvect (ox64);
if (! old-int 64) setvect (ox64,1); /* set a flag */
Else printf (" TSR is already installed")
```

(২) এরপর DOS active flag এর লোকেশন বের করা দরকার। লোকেশন খুঁজে সেখানে একটা far pointer সেট করতে হবে। ফলে সব সময় আমরা জানতে পারব DOS এখন active না inactive অবস্থায় আছে।

```
r.h.ah = ox34;
int86x (ox21, &r, &r, &s);
dos-active=Mk-FP(S.es, r.x.bx);
```

(৩) getvect এর মাধ্যমে যে সমস্ত ইন্টারাপ্টের পরিবর্তন করা দরকার সেগুলোর অ্যাড্রেস সেভ করে রাখতে হবে। যেমন

```
old-int28=getvect (ox28);
```

(৪) setvect এর মাধ্যমে ভেক্টর টেবিলে নতুন ISR এর অ্যাড্রেস বসাতে হবে।

```
setvect (ox28, dos-idle)
```

(৫) keep ফাংশনের মাধ্যমে প্রোগ্রামকে Terminate and stay Resident করতে হবে।

নতুন ISR কেমন করে বানাতে হয় তার উদাহরণ নীচে দেওয়া হলো।

```
void interrupt dos-idle ()
{ (* old-int28) ( ); /* origial routine */
if (!busy) tsr_routine ();
}
```

এখানে busy একটা flag যাকে TSR রুটিন এক্সিকিউট করার সময় সেট (১) করা হয়। ফলে ২য় বার active হওয়ার সুযোগ থাকে না। (চলবে)

*** আহসান হাবীব পলাশ**
কমপিউটার কৌশল, বুয়েট