Процесирање грешака

Емулатор приказује грешке преко посебног прозора. Пример погрешног кода:

```
org 100h
MOV DS, 100
MOV AL, 300
RET
```

MOV DS, 100 је погрешна инструкција пошто сегментни регистри се не могу постављати на одређене вредности директно већ треба радити са регистрима опште намене: MOV AX, 100 MOV DS, AX
MOV AL, 300 је илегална инструкција јер AL регистар има само 8 битова и његова максимална вредност је 255 (11111111b) а минимална -128.

Приликом сејвирања асемблираног фајла, компајлер такође сачува 2 друга фајла који се касније користе од стране емулатора за приказ оригиналног изворног кода када се стартује бинарни егзекутабилан фајл и бирају одговарајуће линије. Често оригинални код се разликује од disabled кода јер не постоје коментари, без сегмената и без декларација промењивих. Компајлерске директиве не производе бинарни код, али све је конвертовано у чист машински код. Понекад једна оригинална инструкција је асембленована у неколико инструкција машинског кода (пример ROL AL, 5 се асемблира у пет секвенцијалних ROL AL, 1 инструкција).

*.~asm овај фајл садржи оригинални изворни код који се користио при креирању егзекутабилног фајла

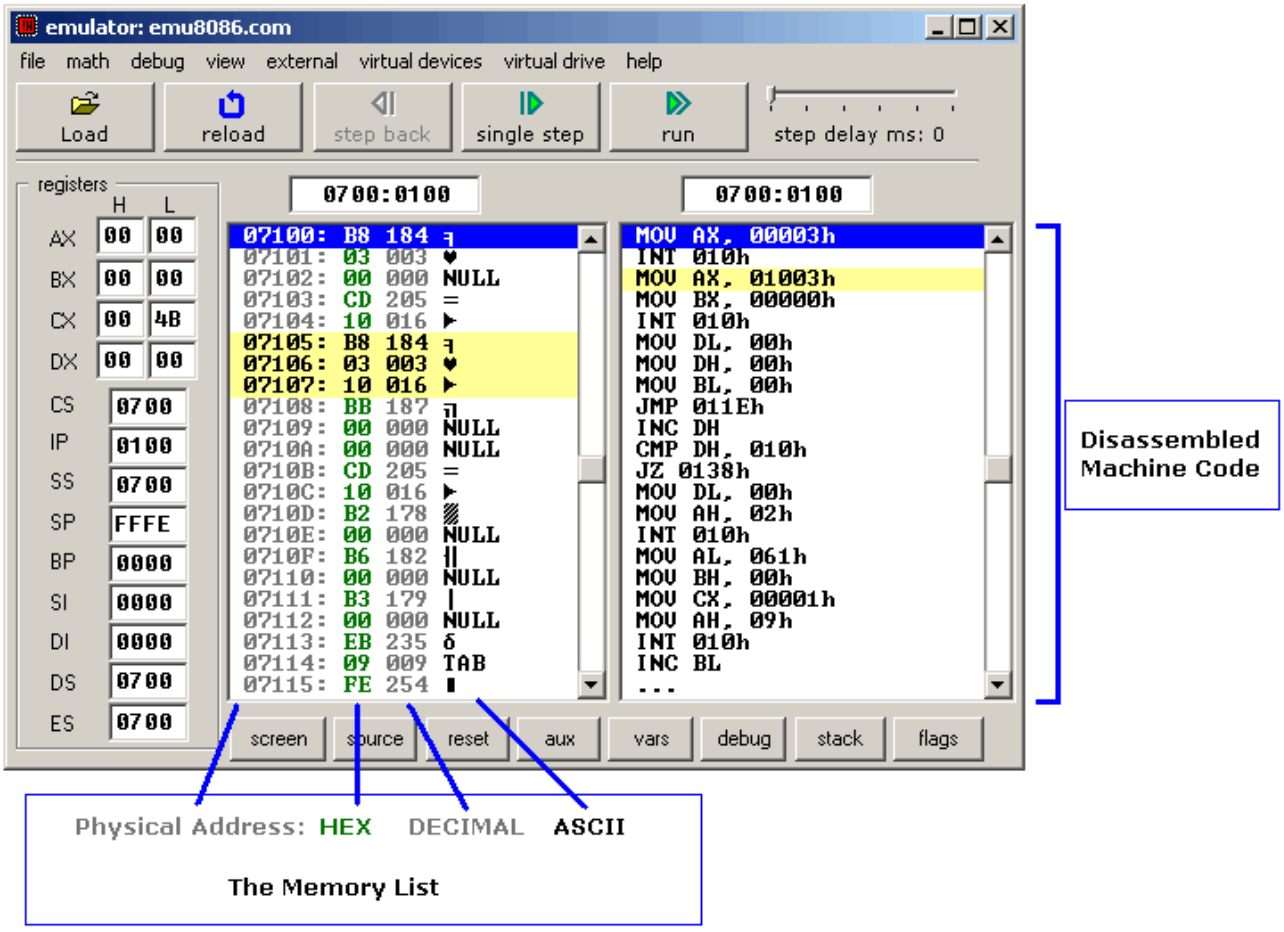
*.debug овај фајл има информације које омогућавају да емулатор бира линије оригиналног изворног кода док извршава машински код

*.symbol табела симбола садржи информације које омогућавају приказ прозора са промењивима

*.binf овај ASCII фајл садржи инфо које је користио емулатор за учитавање bin фајлова са одређене локације и постављање вредности у регистрима пре извршавања (само ако је извршен bin фајл).

Коришћење емулатора

Ако треба учитати код у емулатор, кликне се на emulate дугме. Могуће је учиати извршне фајлове који немају изворне кодове кликом на Load дугме. Емулатор може учитати извршне фајлове направљене другим асемблерима кликом на show emulator из emulator менија. Затим се учитају фајлови из MyBuild или неког другог фолдера. Ако нема фајлова у том фолдеру врати се у едитор изабери Examples учитај било који пример, компајлирај га и затим учитај у емулатор.



Single Step дугме извршава инструкције једну по једну и зауставља се после сваке инструкције.
Run дугме извршава инструкције једну по једну са закашњењем постављеним помоћу step delay између инструкција.
Двоструки клик на регистарски текст-бокс отвара Extended Viewer прозор са вредностима тог регистра конвертованим у свим могућим формама. Могу се променити вредности регистра директно коришћењем овог прозора.
Двоструки клик на меморијску листу отвара Extended Viewer са WORD вредности учитаној из меморијске листе на изабраној локацији. Мање важан бајт је на нижој адреси: LOW BYTE је учитан са изабране позиције а HIGH BYTE са следеће меморијске адресе. Могуће је променити вредности меморијске речи директно у Extended Viewer прозору. Чак је могуће мењати и вредности регистара током извршавања куцањем преко постојећих вредности. Flags дугме даје поглед и мењање флегова током извршавања.
Виртуелни драјвери се сматрају највише четири виртуелна флопи диска FLOPPY_0 до 4.

**Пример01**: Укуцати следећи код и објаснити промене које изазива у меморији процесора.

```
name "ui01";davanje imena fajlu
org 100h     ;kreiranje malog com fajla
mov ax, 03h
int 10h      ;otvara se prozor 80x25 znakova
mov ax, 01h
int 10h      ;otvara se prozor 40x25 znakova
ret          ;vraćanje kontrole OS-u
```

Овим кодом се креира меморијски мало захтеван код који отвара два прозора различитих димензија. Види се да је могуће да у једном моменту само један прозор буде активан и отворен. После извршења сваке појединачне линије кода са леве стране екрана се виде промене у садржајима регистара. Записати све промене које се обележавају плавом бојом.

**Пример02**: Укуцати следећи код и објаснити промене које изазива у меморији процесора.

```
name "ui02"  ;davanje imena fajlu
org 100h     ;kreiranje malog com fajla
mov ah, 02h  ;priprema za ispis u prozoru
mov dl, '1'  ;znak za ispis
int 21h      ;otvaranje prozora za ispis
ret          ;vracanje kontrole OS-u
```

Домаћи задатак 01:
1) Ако се у примеру02 уместо **1** укуца **11** долази до грешке. Објаснити речима какве.
2) Ако се у примеру01 уместо **ax** укуца **ah** на оба места, које вредности ће се појавити у регистру **AX** ?
3) Додати следећи код пре ret линије mov ah, 00h int 16h па објаснити шта се променило на излазу.
4) На излазном прозору треба да се испише **101**. Написати код у асемблеру који то омогућава.
Одговори: 1) Вредност 11 заузима 16 битова а int 21h може да испише само осмобитне вредности.
2) Појавиће се унете вредности у коду као део HIGH BYTE меморијске адресе.
3) После приказа вредности у прозору јавља се андерскор линија која упозорава корисника да се чека унос са тастатуре.
4) Унети у код примера02: mov ah, 02h mov dl, '0' int 21h па поново за вредност 1