מדריך למשתמש - VAX11 Simulator

הקדמה

חבילת התוכנה מורכבת ממספר רכיבים המקושרים ביניהם: עורך טקסט, אסמבלר ודמיין. באמצעות רכיבים אלו אנו מסוגלים לכתוב קוד בשפת האסמבלר של VAX-11, להפוך אותו לקוד בשפת מכונה ולהריץ אותו. כמו כן, סביבת העבודה כוללת כלים להרצת התוכנית צעד אחד צעד ולניפוי שגיאות. הממשק עוצב בדומה לרוב תוכניות Windows הקיימות, על מנת לאפשר למשתמשים להתחיל לעבוד עימו במהירות ובנוחות.

תכונות עיקריות

עורך הטקסט

- כולל Syntax Highlight המותאם לכתיבת קוד עבור VAX-11.
- מאפשר עריכה של מספר קבצים בו זמנית.

אסמבלר

- אסמבלר מתוחכם המספק מידע מפורט במקרה של שגיאות.
- כולל אפשרות לאופטימיזציה של הקוד.
- מאפשר להציג קובץ פלט הכולל את שפת המכונה במקביל לקוד.

דמיין

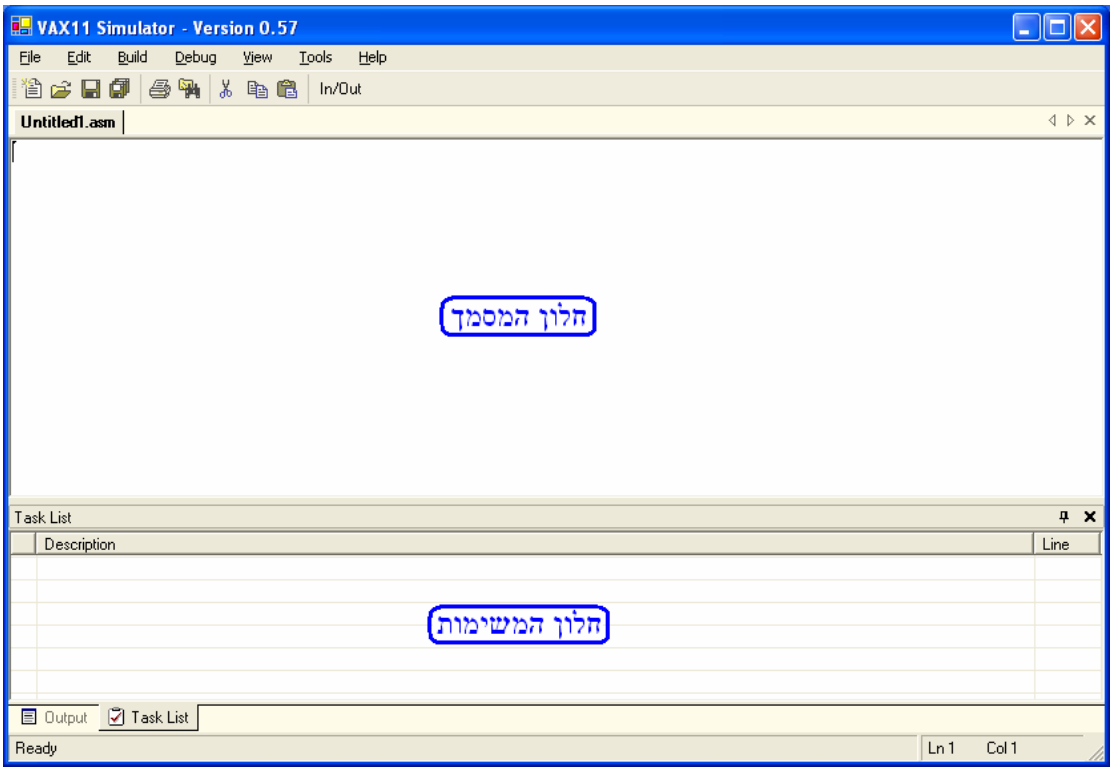
- דמיין התומך ברוב ה-Opcodes של מחשב ה-VAX11
- תומך בעשרות שגרות מערכת של VAX.
- תומך בפסיקות ובחריגות.
- תומך בזיכרון וירטואלי, מאפשר מרחב כתובות של 4GB בתים.
- מכיל הדמיה של זיכרון פיסי ו-Page Faults.
- מאפשר לעקוב אחרי זמן ביצוע התוכנית.

מנפה שגיאות:

- מאפשר הרצת התוכנית שורה אחר שורה, תוך כדי שהסביבה מסמנת את השורה הפעילה.
- תומך בהוספת נקודות עצירה לתוכנית.
- מאפשר לצפות במצב הרגיסטרים, המחסנית והזיכרון בכל רגע.

התחלת העבודה

כאשר נפתח את סביבת העבודה ייפתח החלון הראשי של התוכנית.



מסך הפתיחה של הסימולטור

חלון המסמך הוא המקום בו כותב המשתמש את הקוד של התוכנית. חלון זה מתפקד כעורך טקסט פשוט.

חלון המשימות משמש את האסמבלר. במידה וישנן שגיאות בקוד המשתמש תופיע רשימת השגיאות בחלון זה. כל שגיאה מופיעה בשורה נפרדת, כאשר לחיצה כפולה על שורת השגיאה מקפיצה את חלון המסמך אל השורה המתאימה בקוד.

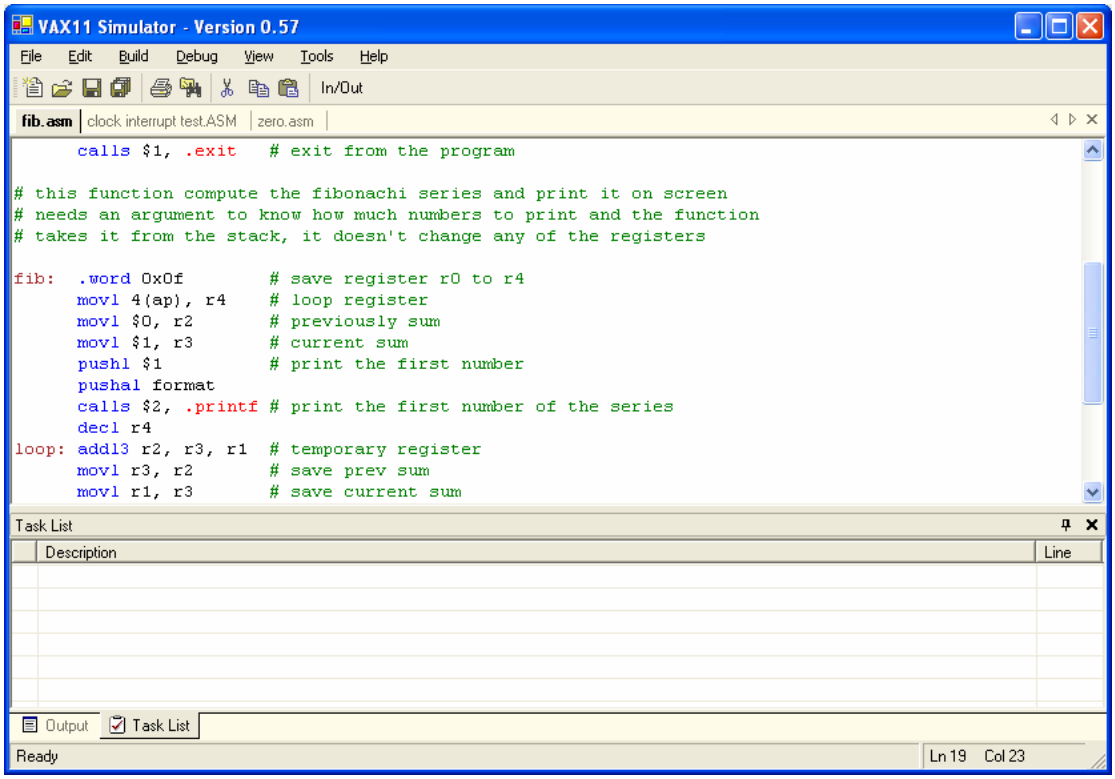
לצד חלון המשימות נמצא **חלון הפלט**, אליו האסמבלר והדמיין שולחים מידע בזמן פעולתם על מצב ההידור או מצב הריצה של התוכנית.

עורך הטקסט

סביבת העבודה מאפשרת למשתמש לערוך קבצי טקסט, בהם ייכתבו תוכניות VAX.

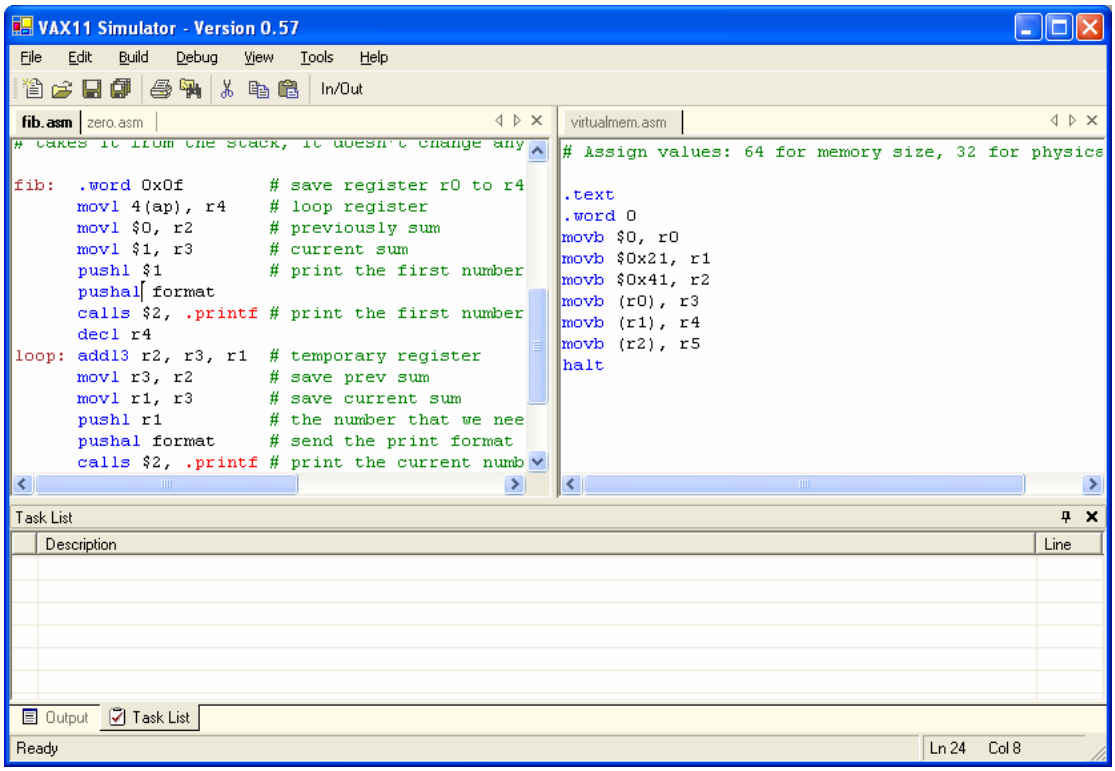
כל **מסמך** (תוכנית) של המשתמש נמצא בחלון מסמך משלו, כאשר ניתן לעבור בין המסמכים השונים הפעילים.

בזמן כתיבת המסמך סביבת העבודה צובעת באופן אוטומטי פקודות הקשורות אל ה-VAX לנוחות המתכנת. הצבעים ניתנים להתאמה אישית מלאה דרך תפריטי סביבת העבודה.



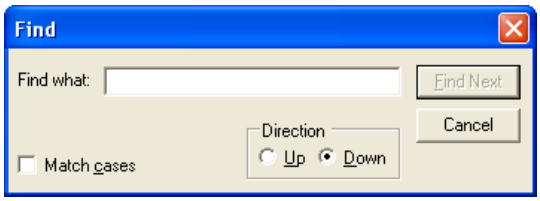
שלושה מסמכים פתוחים

סביבת העבודה מאפשרת גם לפצל את המסמכים הפעילים, כך שכל אחד מהם יתפוס חלק משטח העבודה. פיצול המסמכים נעשה על ידי לחיצה על שם המסמך בסימניית המסמכים, וגרירתו הצידה.



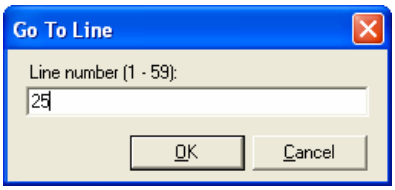
פיצול מסמכים לחלונות מקבילים

עורך הטקסט מאפשר לבצע חיפוש בטקסט. לחיצה על האפשרות Find בתפריט Edit או לחיצה על המקשים Ctrl+F תפתח את חלון החיפוש. ניתן להקליד טקסט ולמצוא את כל המופעים של אותו הטקסט במסמך הפעיל.

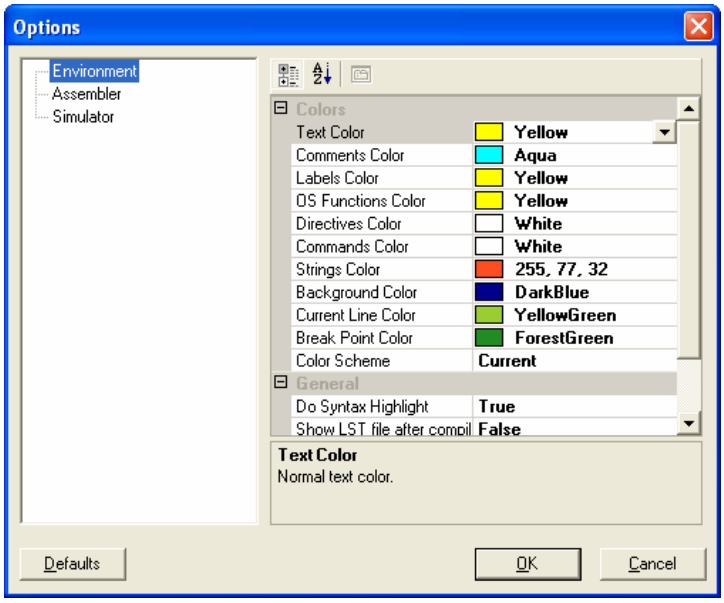


חלון החיפוש

ניתן גם לקפוץ לשורה ספציפית בקובץ, וזאת על ידי בחירת התפריט Go To שנמצא תחת Edit, או על ידי לחיצה על צירוף המקשים Ctrl+G.



חלון האפשרויות



חלון האפשרויות – אפשרויות סביבת העבודה

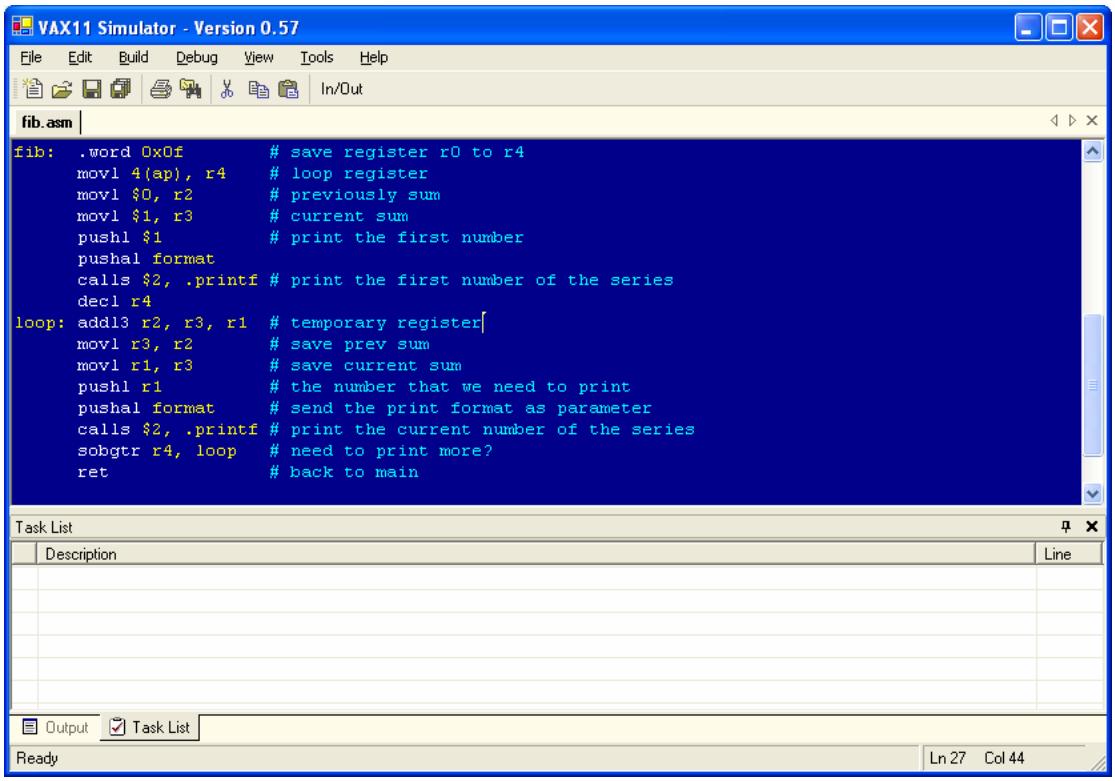
אספקטים רבים בסביבת העבודה, באסמבלר ובסימולטור ניתנים להתאמה אישית עבור המשתמש.

חלון האפשרויות מאפשר לקבוע את ההגדרות השונות עבור סביבת העבודה.

סביבת העבודה

צבעים	
פרמטר	**משמעות**
Text Color	הצבע של כל טקסט שאינו בעל משמעות מיוחדת עבור VAX
Comments Color	צבע ההערות בקוד
Labels Color	צבע התגיות
OS Functions Color	צבע קריאות מערכת
Directives Color	צבע ההנחיות למהדר
Commands Color	צבע ה-opcodes של ה-VAX
Strings Color	צבע קבועי מחרוזות
Background Color	צבע הרקע של המסמך
Current Line Color	צבע השורה הנוכחית (רלוונטי בזמן ניפוי שגיאות)
BreakPoint Color	צבע שורות בהן יש נקודות עצירה
Color Scheme	בחירה של ערכת צבעים מבין הערכות הבאות יחד עם הסימולטור.

כללי	
פרמטר	**משמעות**
Do Syntax Hightlight	קובע האם הטקסט ייצבע או לא
Show LST file after compile	במידה ואפשרות זו נבחרת, לאחר כל הידור מוצלח יוצג קובץ LST המציג את תוצאת ההידור
Show agent on startup	קובע האם מרלין הקוסם יברך את המשתמש עם הפעלת הסימולטור



סביבת העבודה עם צבעים מותאמים אישית

אסמבלר

כללי	
פרמטר	**משמעות**
Optimize Code	במידה ואפשרות זו נבחרת, האסמבלר ייצור את הקוד הקטן ביותר האפשרי עבור קוד המשתמש. על מנת להשוות את האסמבלר הנוכחי לאסמבלר הישן יש לבטל אפשרות זאת, מכיוון שהאסמבלר הישן לא תמך באופטימיזציה של הקוד.
Save LST file after compile	אם אפשרות זו נבחרת, האסמבלר ייצר קובץ פלט עם תוצאת ההידור לאחר כל הידור מוצלח.

סימולטור

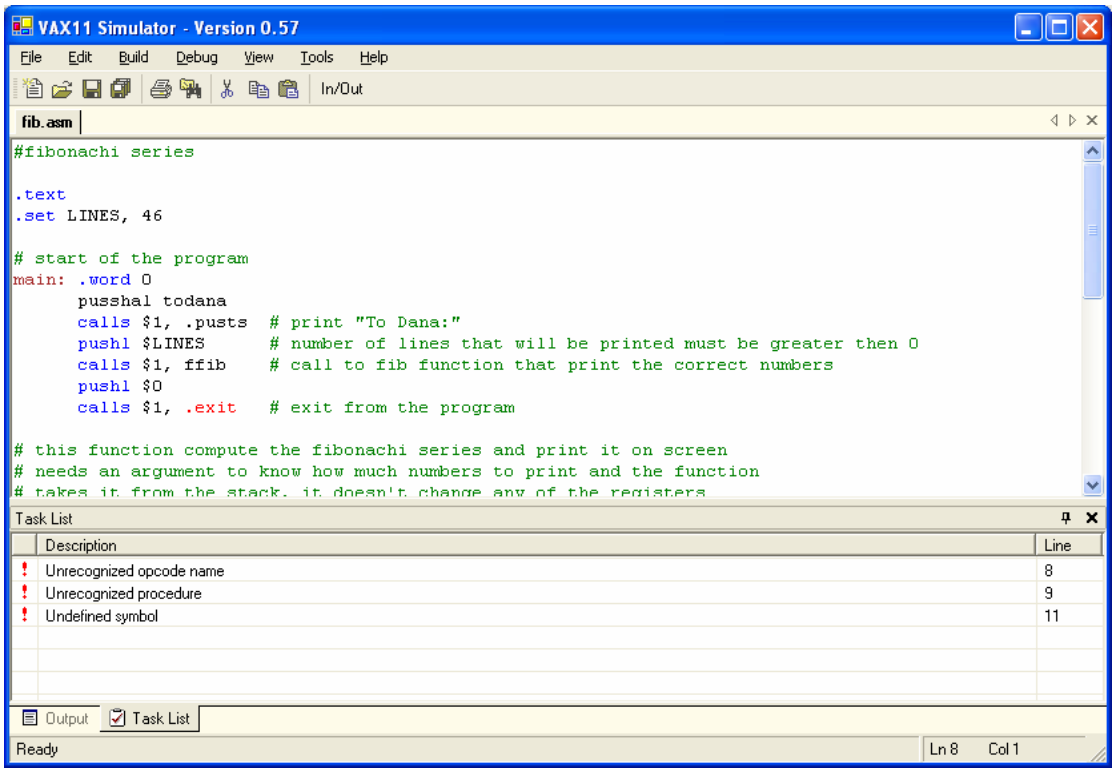
הסימולטור מכיל הדמיה של זיכרון פיסי – המחולק לדפים והתומך בזיכרון וירטואלי. ההגדרות הנוגעות לזיכרון מתייחסות אל ההדמיה הקיימת בסימולטור. ה-Console הוא חלון הקלט/פלט של הסימולטור.

Console	
פרמטר	**משמעות**
Text Color	צבע הטקסט של חלון ה-Console
Background Color	צבע הרקע של חלון ה-Console
Always on top on debug mode	קובע הם חלון ה-Console יהיה Always On Top, אם אנחנו במצב ניפוי שגיאות
כללי	
פרמטר	**משמעות**
Show Registers in Hex	במידה ונבחר, ערכי הרגיסטרים במצב ניפוי יופיעו כמספרים הקסדצימליים
Show Special Registers	במידה ונבחר, רסיגטרים מיוחדים הנגישים בדרך כלל במצב גרעין (kernel) בלבד יוצגו במצב ניפוי
Show Debug Information	במידה ונבחר, הסימולטור ייצר פלט מפורט על מצב הסימולטור במידה ומתרחשת שגיאה הגורמת לסיום התוכנית
זיכרון	
פרמטר	**משמעות**
Page Size	גודל דף בזיכרון של הסימולטור
Physical Memory Size	גודל הזיכרון הפיסי
Show Memory Accesses	במידה ואפשרות זו נבחרת, יוצגו למשתמש כל הגישות לזיכרון שמבצעת התוכנית
Show Physical Addresses	קובע האם להציג את הכתובת הפיסית במקביל לכתובת הוירטואלית כאשר אנו מציגים את הגישות לזיכרון
Show Page Faults	קובע האם להציג הודעה כאשר מתרחש Page-Fault
Fill Memory With Garbage	במידה ונבחר, הזיכרון יתמלא באופן רנדומלי בזבל, ולא באפסים.

האסמבלר

לאחר שנכתוב קוד נוכל להדר אותו דרך התפריט Build ובחירת האפשרות Compile.

במידה והיו שגיאות בקוד, תופיע רשימת השגיאות בחלון המשימות ותינתן לנו אפשרות לתקנן.



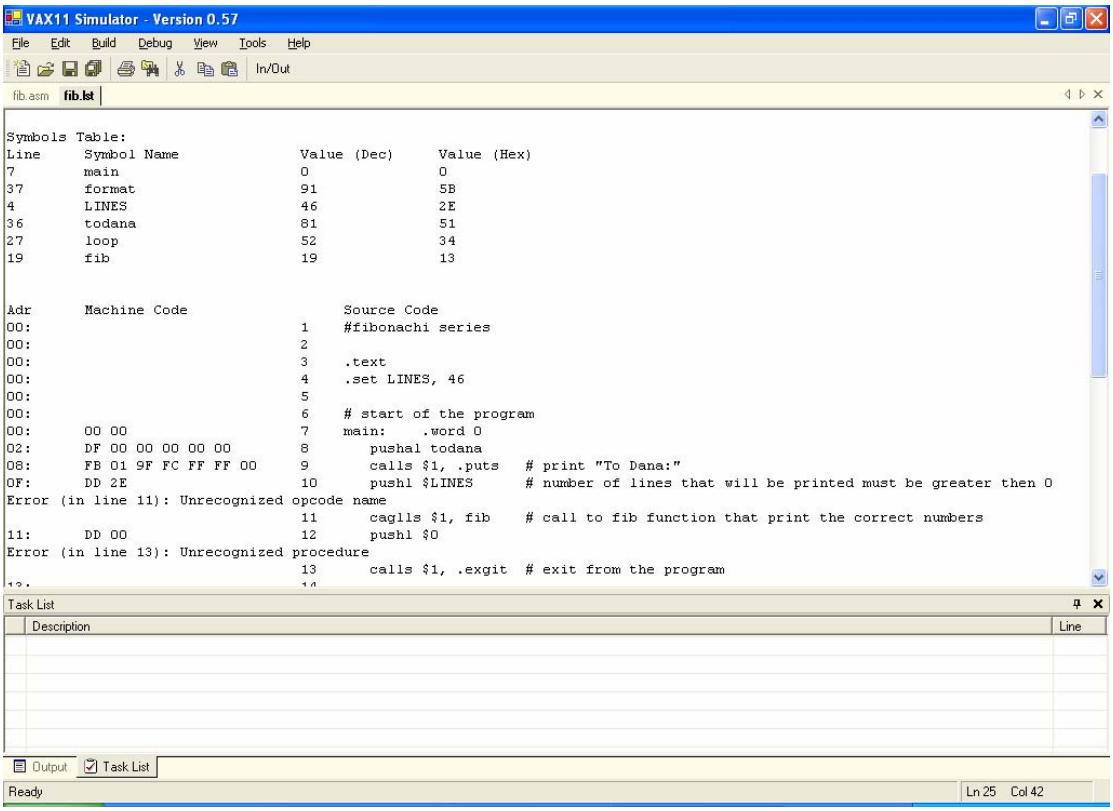
קוד עם שגיאות – השגיאות מופיעות בחלון המשימות

קבצי LST הם קבצים המציגים את קוד המשתמש במקביל לקוד המכונה אותו ייצר האסמבלר.

סביבת העבודה מאפשרת למשתמש להביט ולשמור את קבצי ה-LST.

על מנת לצפות בקובץ ה-LST, נלך לתפריט Build ושם נבחר View LST File.

יש לציין כי ניתן לצפות בקובץ ה-LST גם במידה והיו שגיאות בזמן ההידור, ובמקרה זה יופיעו השגיאות ב-LST לצד הקוד.


VAX11 Simulator - Version 0.57
File Edit Build Debug View Tools Help
In/Out
fib.asm fib.lst
Symbols Table:
Line Symbol Name Value (Dec) Value (Hex)
7 main 0 0
37 format 91 5B
4 LINES 46 2E
36 todana 81 51
27 loop 52 34
19 fib 19 13
Adr Machine Code Source Code
00: 1 #fibonachi series
00: 2
00: 3 .text
00: 4 .set LINES, 46
00: 5
00: 6 # start of the program
00: 00 00 7 main: .word 0
02: DF 00 00 00 00 00 8 pushal todana
08: FB 01 9F FC FF FF 00 9 calls $1, .puts # print "To Dana:"
0F: DD 2E 10 pushl $LINES # number of lines that will be printed must be greater then 0
Error (in line 11): Unrecognized opcode name
11 caglls $1, fib # call to fib function that print the correct numbers
11: DD 00 12 pushl $0
Error (in line 13): Unrecognized procedure
13 calls $1, .exgit # exit from the program
Task List
Description
Line
Output
Task List
Ready
Ln 25 Col 42

LST קובץ

הדמיין

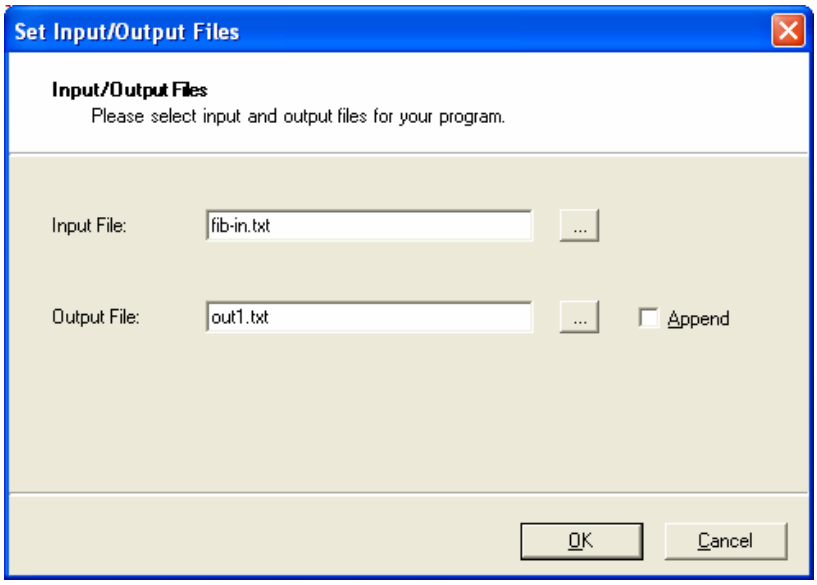
הדמיין מאפשר הרצת תוכניות VAX. הדמיין מחקה אספקטים רבים של החומרה, כולל זיכרון, רגיסטרים, פסיקות, חריגות ועוד.

על מנת להריץ תוכנית שכתבנו, נלך לתפריט Build ונבחר באפשרות Execute. ייפתח חלון Console בו יוצג הפלט של התוכנית שלנו, ובו נוכל להכניס קלט במידה והתוכנית שכתבנו דורשת זאת.

חלון Console עם תוכנית המציגה מספרי פיבונצ'י

הדמיין מאפשר לנו לקבוע קבצי קלט ופלט, על מנת שנוכל לקרוא נתונים מהם במקום מהמקלדת ואל המסך.

על מנת להגדיר קבצי קלט/פלט עבור תוכנית מסוימת, יש ללחוץ על התפריט Build, ושם על Set Input/Output Files ולבחור את קבצי קלט הפלט הרצויים.
יש לשים לב שהגדרת קבצי הקלט-פלט היא אישית עבור כל אחד מהמסמכים הפתוחים.
האפשרות Append מאפשרת להוסיף את הפלט בסוף קובץ קיים.

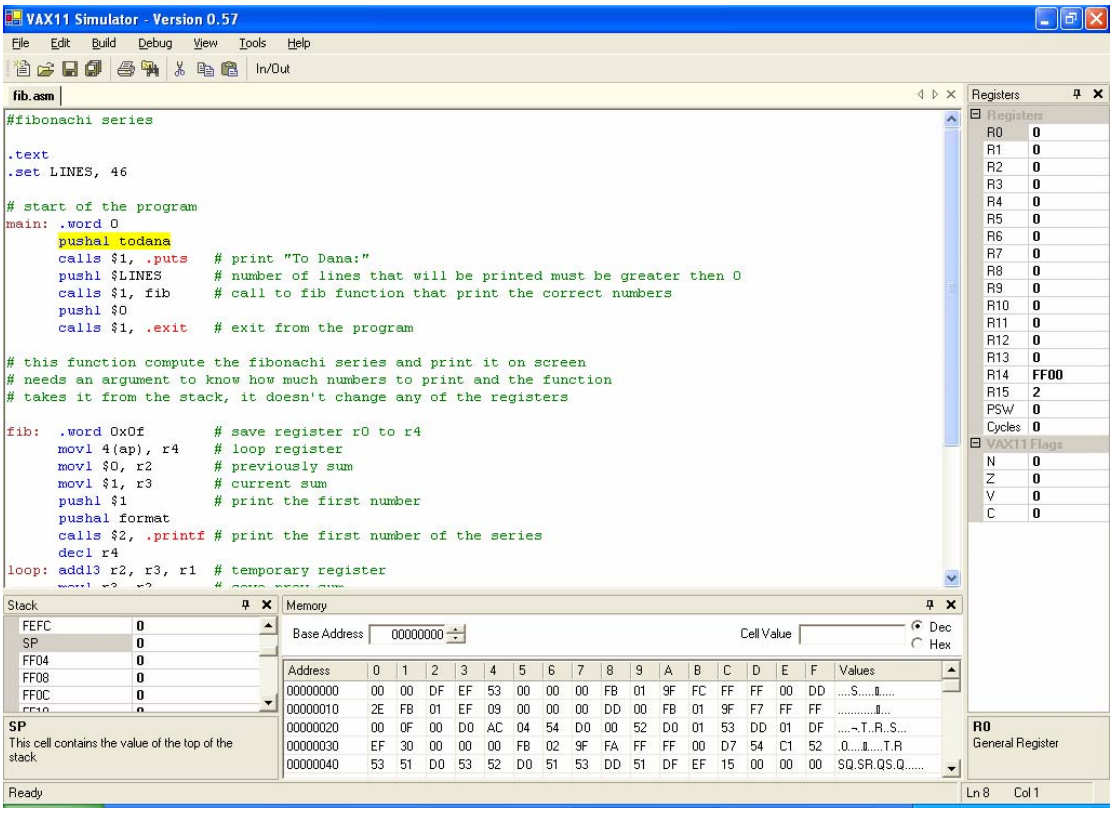


חלון בחירת קבצי קלט/פלט

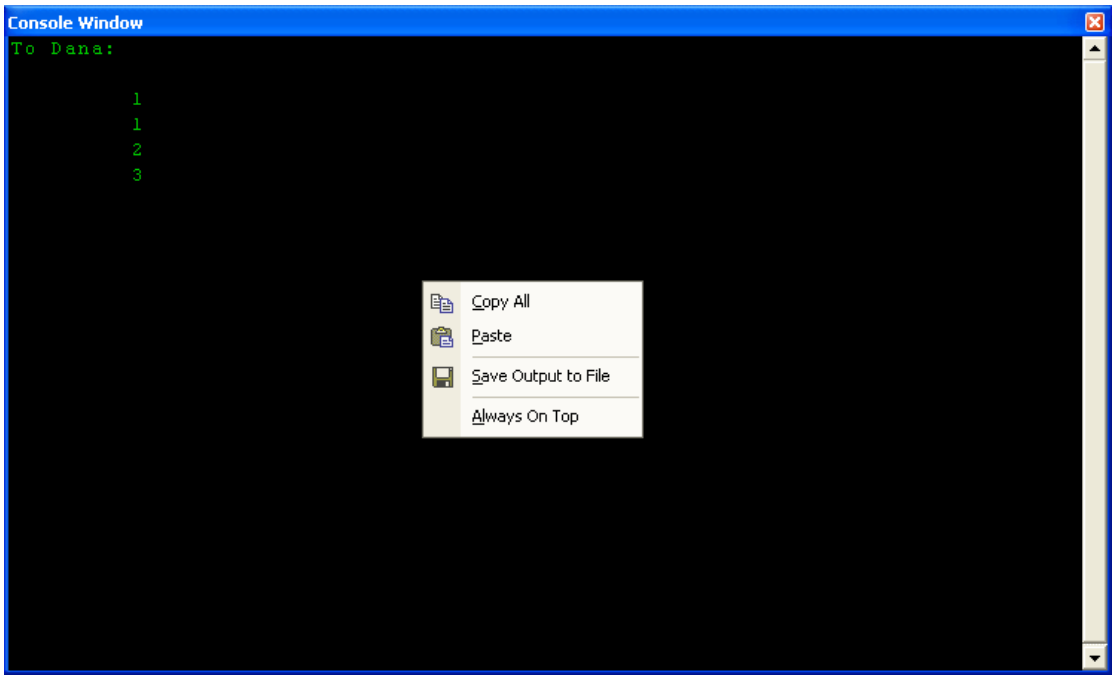
תפריט ה-Debug מאפשר הפעלת התוכנית במצב ריצה צעד אחד צעד, על מנת לתקן שגיאות בתוכנית.

הפעלת התוכנית במצב ריצה צעד אחר צעד נעשה על ידי האפשרות Step בתפריט Debug.

ייפתחו בסביבת העבודה חלונות נוספים, המציגים את מצב המערכת: רגיסטרים, זיכרון ומחסנית. כמו כן, השורה הבאה לביצוע תודגש כל רגע בצבע.



בנוסף ייפתח חלון ה-Console בו יופיע פלט התוכנית בכל רגע ורגע.



אפשרות שימושית נוספת הקיימת בחלון ה-Console היא לחיצה על המקש ימני של העכבר בנקודה כלשהי ב-Console. לחיצה על המקש הימני תפתח תפריט בו מספר אפשרויות: נוכל לקבוע שהחלון יהיה מעל כל החלונות האחרים, נוכל להעתיק את פלט התוכנית שלנו, ונוכל להדביק מידע שישמש כקלט לתוכנית.

התוכנית הראשונה

נציג תוכנית פשוטה המדפיסה על המסך את המחרוזת "Hello, World", וננתח את התוכנית.

```
.text

.org 0x1000
main: .word 0

      # Print a message
      pushal strHello
      calls $1, .puts

      # Exit Program
      pushl $0
      calls $1, .exit

.data

strHello:   .asciz "Hello, World\n"
```

כל תוכנית חייבת להתחיל בהנחיה ".text" המציינת את תחילת קוד התוכנית.

ההנחיה ".org" אומרת לסימולטור למקם את הקוד החל מכתובת המצוינת כפרמטר. במקרה שלנו הקוד ימוקם החל מהכתובת 0x1000. אין כל הגבלה למקם את התוכנית בכל מקום בזיכרון, אם כי לא מקובל למקם את התוכנית בכתובות הנמוכות, עקב העובדה שבמערכת אמיתית מערכת ההפעלה היא הנמצאת בכתובות הזיכרון הנמוכות.

התגית main היא נקודת ההתחלה של התוכנית. הסימולטור מחפש תמיד את תגית זו, ובמידה ונמצאה היא משמשת כנקודת ההתחלה של התוכנית. במידה ונרצה להתחיל את התוכנית בנקודה אחרת ולא ב-main נשתמש בהנחיה ".entrypoint".

לאחר התגית main מופיעה מילת המסכה, ולאחריה קוד התוכנית.

כל תוכנית המסתיימת בהצלחה מסתיימת בשורות:

```
      pushl $0
      calls $1, .exit
```

שורות אלו מבצעות קריאה לשגרת מערכת שתפקידה לסיים את התוכנית.

התגית ".data" מציינת את התחלת אזור הנתונים של התוכנית.