emuARM

USER MANUAL
LIST OF CONTENTS

1. Introduction .............................................................................................................................................. 1
2. Menu Bar .................................................................................................................................................. 1
  2.1 File Menu ............................................................................................................................................... 2
    2.1.1 New ............................................................................................................................................... 2
    2.1.2 Open ............................................................................................................................................. 2
    2.1.3 Save ............................................................................................................................................. 3
    2.1.4 Exit ............................................................................................................................................... 4
2.2 Edit Menu ................................................................................................................................................ 4
  2.2.1 Cut .................................................................................................................................................. 4
  2.2.2 Copy ............................................................................................................................................... 4
  2.2.3 Paste .............................................................................................................................................. 4
2.3 Run Menu .............................................................................................................................................. 6
  2.3.1 Run Code ...................................................................................................................................... 6
  2.3.2 Step Into ....................................................................................................................................... 7
  2.3.3 Step Out ....................................................................................................................................... 8
  2.3.4 Step Over .................................................................................................................................... 9
  2.3.5 Reset .......................................................................................................................................... 10
2.4 Help Menu .......................................................................................................................................... 11
  2.4.1 About Us .................................................................................................................................... 11
  2.4.2 Contact Us ................................................................................................................................. 11
  2.4.3 Report a Bug ............................................................................................................................... 12
3. Toolbar .................................................................................................................................................... 12
4. Tree View Display ................................................................................................................................ 12
5. Breakpoints ........................................................................................................................................... 13
6. Text Editor ............................................................................................................................................. 14
7. Register Set .......................................................................................................................................... 15
8. Memory Table ..................................................................................................................................... 16
9. Jump Table .......................................................................................................................................... 17
10. Status Window .................................................................................................................................. 18
1. INTRODUCTION

In today's world, there is an ever increasing array of ARM based mobile devices available in the market, emuARM (Advanced RISC Machine device emulator) is an extremely useful tool available to mobile application developers. The emulator provides a way to test your mobile application on a variety of devices and platforms even if one does not own a physical device. While using one or more real devices can be a valuable aspect of a testing plan, by using the emulator, the developer can verify his application’s behaviour across a much wider variety of virtual devices and configurations than would normally be practical using physical devices. Testing an application on the emulator is faster, easier, and more convenient than with physical devices because one can easily reproduce consistent testing scenarios.

The front end of emuARM is shown below.

All the operations and components of emuARM are described in the following sections.

1. MENUBAR

The menu bar contains all the options and commands that the user will need in the operation of this emulator from basic file management commands to debugging facilities.
2.1 FILE MENU

This option contains the basic File Management tools for the emulator.

2.1.1 NEW

It is used for creation of a new tab for the text editor where one can enter and edit the code. It can also be used by the shortcut Ctrl + N.

The new tab will be named Untitledx.s by default, where x is the number of this tab.

As shown, a new tab opens up by the name of Untitled1.s.

2.1.2 OPEN

Open is used to open an existing file.

After clicking on the Open menu option or pressing Ctrl + O from the keyboard, a dialogue box appears by which we can navigate through the file system and select the necessary file to open.

After clicking on Open, we can now view and edit the required file in the text editor window.
2.1.3 SAVE

This option can act in two ways. The shortcut for both is Ctrl + S.

If the file to be saved is an Untitled.s file, it acts as a Save As option. A dialog box appears as shown, and the user can enter the desired name and click on Save.

Otherwise, if the file has been saved previously, it acts like a Save button and commits the changes that were made to the file to the memory.
2.1.4 EXIT

On selecting this option or on pressing Ctrl + Q, the user exits from the emulator and the window closes.

2.2 EDIT MENU

This menu has the features for editing the text in files.

2.2.1 CUT

Cut is used to crop the selected portion of the text.

It can be done by selecting the text and then either clicking on Cut menu option or pressing Ctrl + X from the keyboard.

This text can then be placed elsewhere using the Paste option.

2.2.2 COPY

Copy is used to only copy the selected portion of the test and place it in the clipboard to be pasted elsewhere. The original text remains at its place.

Shortcut for this is Ctrl + C.

2.2.3 PASTE

This places the text which was most recently cut or copied to be placed at the location of the cursor.

Shortcut for this option is Ctrl + V.

These three operations are explained below.

1. Select text and click on Copy.
   
   The selected text on line 6 is shown in grey colour.
2. Click on Cut if this option is required which would result into the following.

```
   .file   "prime.c"
   .global __mods13
   .section .rodata
   .align 2
   .LC0:    .ascii  "Cannot say\000"
   .align 2
   .LC1:    .ascii  "It is not prime\000"
   .align 2
   .LC2:    .ascii  "It is prime\000"
   .text
```

Here line 6 has been cropped.

3. Paste is used to place this text on line 5.

```
   .file   "prime.c"
   .global __mods13
   .section .rodata
   .align 2
   .LC0:    .ascii  "Cannot say\000"
   .align 2
   .LC1:    .ascii  "It is not prime\000"
   .align 2
   .LC2:    .ascii  "It is prime\000"
   .text
```

...
2.3 RUN MENU

This menu has the options for executing and debugging the programs.

![Image of Run Menu]

2.3.1 RUN CODE

This option is used to execute the whole program and show the output in the status window, reflect the changes in the register set, memory table, jump table, whichever is applicable.

The shortcut for the same is F6.

Example: The following code for conditional execution of the AND instruction is run.

```
1  mov r2, #2
2  mov r4, #3
3  mov r0, #11
4  cmp r0, #10
5  adds r1, r2, r4
```

The register set shows the following values.

```
<table>
<thead>
<tr>
<th>Register Set</th>
<th>Memory Table</th>
<th>Jump Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>11</td>
<td>R1</td>
</tr>
<tr>
<td>R2</td>
<td>2</td>
<td>R3</td>
</tr>
<tr>
<td>R4</td>
<td>3</td>
<td>R5</td>
</tr>
<tr>
<td>R6</td>
<td>0</td>
<td>R7</td>
</tr>
<tr>
<td>R8</td>
<td>0</td>
<td>R9</td>
</tr>
<tr>
<td>R10</td>
<td>0</td>
<td>R11</td>
</tr>
<tr>
<td>R12</td>
<td>0</td>
<td>R13</td>
</tr>
<tr>
<td>R14</td>
<td>-1</td>
<td>R15</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>V</td>
</tr>
</tbody>
</table>
```
2.3.2 STEP INTO

This is a debug mode which executes the code one line at a time sequentially and we can view the results of executing till that line simultaneously in the register set or wherever applicable. The currently executing line is highlighted for ease of reference.

We can also debug only a portion of the code using breakpoints.

After clicking on Step Into, a Next button appears adjacent to the toolbar.

Clicking on this will take the user to the first line to be executed, after which every click will correspond to the execution of the subsequent line.

Corresponding output till this line (line number 2) is as follows.

```
1  mov r2, #2
2  mov r1, #3
3  mov r0, #11
4  cmp r0, #10
5  adds r1, r2, r4
```
When the user reaches the last line of the code(or the portion of the code) to be debugged, the Next button and the other debugging modes are deactivated and a warning “End of File reached”, appears to the right of the Next button. There is a cross button next to this warning that will take the user out of the debug mode.

Also, as soon as the Step Into button is clicked, the other debug modes (Step Out, Step Over) get activated. The pane for setting breakpoints also gets activated as shown below.

The operation of these will be explained in the subsequent sections.

The shortcut for this command is Ctrl + R.

2.3.3 STEP OUT

The Step Out debug mode returns from a function which has been stepped into. Even though we return from the function, the remainder of the code inside the method will be executed normally.

Therefore when the user is executing a line within a function and Step Over is clicked, it brings the user to outside the function with only one click of the Next button. The subsequent lines of the code in the function are executed in one go. Also, if this command is invoked when the current line to be debugged is a function call, the definition of the function is executed in one go and the user is taken to the line next to the function call. In all other circumstances it behaves like Step Into.

Shortcut for this command is F8.
This mode gets activated only when the program is in debug mode, i.e. Step Into is clicked on.

At line 33, Step Out is called at which point it executes the rest of the lines of the function normally and reaches line 25 (the line after the function call) as shown in the second figure.

24     bl    fn
25     mov   r0, r3
26     b      hi
27
28    fn:
29         @ args = 0, pretend = 0, frame = 0
30         @ frame_needed = 1, uses_anonymous_args =
31         mov   ip, sp
32         stmfd  sp!, {fp, ip, lr, pc}
33         sub   fp, ip, #4
34     ldr    r0, .L2
35     bl     printf
36     ldmfd  sp, {fp, sp, pc}

23     sub    fp, ip, #4
24     bl     fn
25     mov   r0, r3
26     b      hi
27
28    fn:
29         @ args = 0, pretend = 0, frame = 0
30         @ frame_needed = 1, uses_anonymous_args = 0
31         mov   ip, sp
32         stmfd  sp!, {fp, ip, lr, pc}
33         sub   fp, ip, #4
34     ldr    r0, .L2
35     bl     printf
36     ldmfd  sp, {fp, sp, pc}

2.3.4 STEP OVER

Step Over command is used for stepping over the next function call (without entering it) at the currently executing line of code. Even though the method is never stepped into, the method will be executed normally.

The shortcut for this is F9.

This mode too gets activated only when the program is in debug mode.

The following figures explain the functioning of this debug mode.
On clicking on Step Over at the function call on line 24, we execute the definition of the function normally and reach line 25 as seen below.

```assembly
23  sub  fp, ip, #4
24  bl  fn
25  mov  r0, r3
26  b   hi

28 fn:
29  @ args = 0, pretend = 0, frame = 0
30  @ frame_needed = 1, uses_anonymous_args = 0
31  mov  ip, sp
32  stmfd sp!, {fp, ip, lr, pc}
33  sub  fp, ip, #4
34  ldr  r0, .L2
35  bl   printf
36  ldmfd sp, {fp, sp, pc}

2.3.5 RESET

This option is used for resetting the breakpoints in the breakpoint panel such that the red colored circles which correspond to the selected lines of the program are now cleared and reset.

The shortcut for this is F10.

After resetting the breakpoints, the breakpoint panel is cleared as shown.
2.4 HELP MENU

The help menu lists general options for information about the emulator and the developer, and contact details.

2.4.1 ABOUT US

This displays basic details about the emulator as shown below.

2.4.2 CONTACT US

The user can contact the developer for any queries at emuarmqueries@gmail.com.
2.4.3 REPORT A BUG

In case of any errors in the emulator, the user can contact the developers of this emulator at emuarmqueries@gmail.com.

![Message] (Thank you for your report. Please mail the details to emuarmqueries@gmail.com)

2. TOOLBAR

The toolbar contains shortcuts for various commands that are available in the menu bar.

![Toolbar](Image)

The icons represent the following actions.

- **New**
- **Open**
- **Save**
- **Exit**
- **Cut**
- **Copy**
- **Paste**
- **Run Code**
- **Step Into**
- **Step Out**
- **Step Over**

The operation of these has been explained in the previous sections.

3. TREE VIEW DISPLAY

This panel appears on the left side of the emulator and displays the file structure of the computer in a hierarchical format. The root of the tree view points to the home directory. Any existing file can be opened in the text editor through this tree view.

A scroll bar also appears for navigation through this tree view.

Double-clicking on a folder will display its structure of files and folder while single-clicking on a file will open it in the text editor.
4. BREAKPOINTS

Breakpoints are used for limiting the code to be debugged to a specific portion and can be set through the breakpoint panel.

If the user clicks on the panel, adjacent to the line number for which he wants to set the breakpoint, a red circle appears. Only placing the cursor on the panel produces a tool tip with the line number corresponding to the line in the code.

The user can thusly set either one breakpoint, in which case the first line of the program will act as the initial line of the portion to be debugged and the selected line (through the breakpoint) will act as the terminating line, or we can set two breakpoints where the first will act as the initial line and the second will act as the final line of the portion of the code to be debugged.

The breakpoints can be reset with the reset command.
The following sequence shows the operation of breakpoints. As shown, as the debugging finishes, the breakpoints are reset automatically.

5. TEXT EDITOR

This is the place where the user enters the program or views an existing program. Multiple tabs can be opened and closed. Line numbers are also displayed.
6. REGISTER SET

This displays the values in the register set which contains the following 32 bit registers:

- r0-r12 general purpose registers
- r13 (the stack pointer) and r14 (link register)
- r15 (the program counter)
- cpsr (current program status register)
- spsr (saved program status register)

The program status registers have the following condition codes.

N = Negative result from ALU flag.
Z = Zero result from ALU flag.
C = ALU operation Carried out
V = ALU operation Overflowed
The values in the register set are displayed in the decimal format by default. By right clicking anywhere on the register set, we can set the format to decimal, hexadecimal, binary and octal.

For example, these following values are shown in decimal format (default). We right click and select required format.

<table>
<thead>
<tr>
<th>R0</th>
<th>Decimal</th>
<th>R1</th>
<th>2718</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>436</td>
<td>R3</td>
<td>987</td>
</tr>
</tbody>
</table>

The following figure shows the values in hexadecimal representation.

<table>
<thead>
<tr>
<th>R0</th>
<th>1a</th>
<th>R1</th>
<th>a9e</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>1b4</td>
<td>R3</td>
<td>3db</td>
</tr>
</tbody>
</table>

7. MEMORY TABLE

This table displays all the locations at which values are stored in the memory and the respective contents.

Initially the table is empty and progressively augments as a memory location with its respective memory content is added.

These two cases are shown in the following figures.
8. JUMP TABLE

This table refers to the one which stores all the labels and the respective line numbers occurring in the program. It is the third tab in the right side panel of the emulator.

Again as with the memory table, the jump table is initially empty, as shown below.

This is filled at the time of compilation and required number of rows are filled up and added to the table. This is shown in the following figure.
9. STATUS WINDOW

This is the window which displays the statuses after completion of various actions.

The status window when the emulator is started is as shown below and corresponds to the opening of a new untitled file in the text editor.

```
Opening Blank File..
New blank file opened.
```

Various such statuses for saving a program, building a program etc. are displayed here. In case of errors of various kinds in entering the program, the respective status describing the status is displayed.

Example: The following statement is erroneous as the # is missing at the front of the operand.

```
mov r3, 2
```

```
Opening Blank File..
New blank file opened..
Compiling source file run:
ERROR in line 1: Unrecognized operand 2ERROR in line 1:2 is not a valid operand.BUILD FAILED
```
The status window is also where the user can enter input and the output is printed.

Example: For entering two values into registers R0 and R1, the user can write the following line.

```c
scanf("%d,%d", &r0, &r1)
```

On executing this line, the user can position the cursor as required and enter the values separated by a space. After the last value, Enter key is pressed.

```
Opening Blank File.
New blank file opened..
Compiling source file
run: 3 4
BUILD SUCCESSFUL
```

The new values are reflected in the required registers.

<table>
<thead>
<tr>
<th>R0</th>
<th>3</th>
<th>R1</th>
<th>4</th>
</tr>
</thead>
</table>

Statements can also be printed. The following figure shows the printing of “Hello World”.

```
Opening Blank File..
New blank file opened..
Saving File...
Compiling source file C:\Users\namital\Desktop\Sample Files\hello.s
run :
Hello worldBUILD SUCCESSFUL
```