

The screenshot shows the CPUulator ARMv7 System interface. The assembly window displays the following code:

```
00000000  e52db004  push {fp}
00000004  r1           r11
00000008  e1a0b00d  mov    fp, sp      // make the base pointer
0000000c  r1           r11, sp
00000010  e24dd004  sub    sp, sp, #4 ; 0x4 // help rod variables
00000014  e59b0008  ldr    r0, [fp, #8] // load n disks
00000018  e3500001  cmp    r0, #1      // check if n == 0
00000020  e59b2010  bne    r0, [r1, #8] // B if_end
00000024  ea000018  b     0x8c (0x8c: if_end)
00000028  e59b000c  ldr    r0, [r1, #12] // r1 = fromPeg
0000002c  e2600006  ldr    r2, [fp, #16] // r2 = toPeg
00000030  e59b2010  ldr    r1, [r1, #12]
00000034  e59b2010  ldr    r2, [r1, #16]
00000038  e59b2010  ldr    r2, [r1, #16] // we would print this
0000003c  ea000018  b     0x8c (0x8c: if_end)
00000040  e59b000c  ldr    r0, [r1, #12]
00000044  e2600006  ldr    r0, [r1, #16] // r0 = 6 - from_rod
00000048  e59b2010  ldr    r1, [fp, #16] // to_rod
```

The Registers window shows the following values:

r0	00000000
r1	00000000
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000
pc	00000098
cpsr	0000001d3 NZCVI SVC
pspr	00000000 NZCVI ?
s0	00000000
s1	00000000

The Devices section shows various peripherals: LEDs, Switches, Push buttons, Seven-segment displays, and JTAG UART.

For when $n = 3$

The screenshot shows the CPUulator ARMv7 System interface. The top navigation bar includes tabs for Discord, Teams and Channels, CS255 Syllabus, Grades for Ahmed A Ad, Tower Hanoi built - Go!, and CPUulator ARMv7 System. The main window has tabs for Registers, Disassembly (Ctrl-D), Memory (Ctrl-M), and Messages.

Registers tab: Shows the ARM寄存器 (r0-r15, sp, lr, pc, cpsr, spsr) and their values. The PC register is highlighted with a yellow background.

Disassembly (Ctrl-D) tab: Displays assembly code from address 0000000c to 0000009c. The code includes instructions like LDR, PUSH, ADD, SUB, and MOV, along with comments explaining operations like loading from n-disk and performing n-1. The instruction at address 0000008c is highlighted with a yellow background.

Devices section: Shows various peripherals and their states. It includes:

- LEDs:** Eight green LEDs labeled fff00000.
- Switches:** Six switches labeled 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. The switch labeled 'All' is checked.
- Push buttons:** Six push buttons labeled 3, 2, 1, 0. The buttons labeled 'All' and '0' are checked.
- Seven-segment displays:** Four displays showing the value '8'.

JTAG UART section: Shows the serial port activity with Read FIFO: 0 and Write FIFO: 0.

Messages section: Displays compiler output indicating the code was compiled successfully.

From 1 to 3

The screenshot shows the CPUulator ARMv7 System interface. The assembly window displays the following code:

```
r0 00000002  
r1 00000001  
r2 00000002  
r3 00000000  
r4 00000000  
r5 00000000  
r6 00000000  
r7 00000000  
r8 00000000  
r9 00000000  
r10 00000000  
r11 ffffffa4  
r12 00000000  
sp ffffffa8  
lr 00000005c  
pc 0000006c  
cpsr 600001d3 ZC I SVC  
spsr 00000000 NZCV1 ?  
  
● Address    Opcode    Disassembly  
00000050 e2400001 38 SUB r0, r0, #1    // perform n - 1  
00000054 e52d0004 39 sub    r0, r0, #1 ; 0x1  
41 BL hanoi  
00000058 ebffffe8 b1 0x0 (0x0: hanoi)  
42 ADD sp, sp, #12    // Pop parameters  
add    sp, sp, #12 ; 0xc  
46 LDR r1, [fp, #12]    // print move disk n to to_rod  
00000060 e59b100c ldr    r1, [r1, #12]  
47 LDR r2, [fp, #16]  
ldr    r2, [r1, #16]  
49 // do the same thing for (n-1, help_rod, to_rod)  
51 LDR r0, [fp, #16]    // load to_rod  
52 push {r0}  
00000068 e59b0010 54 LDR r0, [fp, #4]    // load help_rod  
0000006c e52d0004 55 ldr    r0, [r1, #4]  
56 push {r0}  
00000070 e51b0004 57 LDR r0, [fp, #8]    // load in n-disk  
00000074 e52d0004 58 ldr    r0, [r1, #8]  
00000078 e59b0008 59 SUB r0, r0, #1    // perform n - 1  
00000080 e2400001 60 sub    r0, r0, #1 ; 0x1  
00000084 ebffffdd 61 BL hanoi  
00000088 e28dd00c 62 ADD sp, sp, #12    // Pop parameters  
add    sp, sp, #12 ; 0xc  
64 if end:  
  
Registers    Call stack    Trace  
Breakpoints    Watchpoints    Symbols  
Counters  
Settings  
Number Display Options  
Size: Hexadecimal  
Format: Hexadecimal  
Memory rows per row: 4  
Messages  
Compiling...  
Code and data loaded from ELF executable into memory. Total size is 184 bytes.  
Assembler: arm-eabi-as -mfloat-abi=softfp -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/arm7MnZDU.s.o work/arm7MnZDU.o  
Link: arm-eabi-ld --script build_arm.ld -e _start -u _start -o work/arm7MnZDU.self work/arm7MnZDU.o  
Compile succeeded.
```

From 1 to 2

The screenshot shows the CPUulator ARMv7 System interface. The assembly window displays the following code:

```
r0 00000001  
r1 00000003  
r2 00000002  
r3 00000000  
r4 00000000  
r5 00000000  
r6 00000000  
r7 00000000  
r8 00000000  
r9 00000000  
r10 00000000  
r11 fffffc34  
r12 00000000  
sp ffffffc30  
lr 00000088  
pc 00000020  
cpsr 000001d3 ZC I SVC  
pspr 000001d3 ZC I SVC  
  
Registers Call stack Trace  
Breakpoints Watchpoints Symbols  
Counters  
Settings  
Number Display Options  
Size: Word Format: Hexadecimal Memory words per row: 4  
Messages  
Compiling...  
Code and data loaded from ELF executable into memory. Total size is 184 bytes.  
Assembler: arm-eabi-as -mfloat-abi=softfp -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmmTilkV.s.o work/asmmTilkV.c  
Link: arm-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmmTilkV.self work/asmmTilkV.s.o  
Compile succeeded.
```

The Registers panel shows the state of various registers. The Devices panel shows the state of LEDs, switches, push buttons, seven-segment displays, and JTAG UART. The assembly window shows the disassembly of the program, with some instructions highlighted in yellow. The bottom status bar indicates the current assembly and memory editors.

From 3 to 2

The screenshot shows the CPUulator ARMv7 System interface. The top navigation bar includes tabs for Discord, Teams and Channels, CS255 Syllabus, Grades for Ahmed A Ad, Tower Hanoi built - Go!, and CPUulator ARMv7 System. The main window has tabs for Registers, Disassembly (Ctrl-D), and Memory (Ctrl-M). The Registers tab shows a list of registers with their current values. The Disassembly tab displays assembly code for the program, with some instructions highlighted in yellow. The Memory tab shows memory dump options. On the right side, there are sections for Devices (LEDs, Switches, Push buttons, Seven-segment displays) and Interrupts (JTAG UART, Cortex-A9 Private Timer, Cortex-A9 Watchdog Timer, HPS L4 Watchdog Timer, Interval Timer). The bottom status bar shows the current assembly address and memory dump range.

From 1 to 3

The screenshot shows a debugger interface with multiple tabs at the top: Discord, Teams and Channels, CS255 Syllabus, Grades for Ahmed A Ad, Tower Hanoi built - Go!, CPUulator ARMv7 System, and a blank tab. The main window has tabs for Registers, Disassembly, and Memory.

Registers Tab: Shows CPU registers r0-r12, sp, lr, pc, cpsr, and spsr. The pc register is highlighted in red, showing the value 00000024.

Disassembly Tab: Displays assembly code for the program. The assembly code includes instructions like andeq, push, mov, sub, ldr, cmp, bne, and b. It also includes comments explaining the purpose of each instruction, such as saving return addresses and frame pointers, making base pointers, and loading disk counts.

Memory Tab: Shows a dump of memory starting at address 00000000. The dump includes sections for LEDs, Switches, Push buttons, Seven-segment displays, and JTAG UART, each with their respective binary values.

Messages Tab: Displays the message "Compiling..." and "Code and data loaded from ELF executable into memory. Total size is 184 bytes." followed by the output of the assembly command.

From 2 to 1

Discord | #learn_tin | Teams and Channels | CS255 Syllabus | Grades for Ahmed A Adi | Tower Hanoi built - Go! | CPUlator ARMv7 System

cpulator.01xz.net/?sys=arm-de1soc

Stopped Step Into F2 Step Over Ctrl+F2 Step Out Shift+F2 Continue F3 Stop F4 Restart Ctrl+R Reload Ctrl+Shift+L File Help

Registers

r0	00000001
r1	00000002
r2	00000003
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	fffffc4c
sp	00000000
lr	fffffc44
pc	00000074
cpsr	600001d3 ZC I SVC
spsr	600001d3 ZC I SVC

Call stack **Trace** **Breakpoints** **Watchpoints** **Symbols** **Counters** **Settings**

Number Display Options
Size: Word Format: Hexadecimal Memory words per row: 4

Messages
Compiling...
Code and data loaded from ELF executable into memory. Total size is 184 bytes.
Assemble: arm-eabi-as -mfloat-abi=softfp -march=armv7-a -mpcu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmmTilkV.s.o work/asmmTilkV.s
Link: arm-eabi-ld --script build_arm.ld -e _start -u _start -u work/asmmTilkV.s.elf work/asmmTilkV.s
Compile succeeded.

</> Disassembly (Ctrl-D)

Go to address, label, or register: 00000000 Refresh

Address	Opcode	Disassembly
00000054	e52d08004	39 push {r0}
00000058	ebffffe8	41 BL hanoi
0000005c	e28dd00c	42 ADD sp, sp, #12 // Pop parameters
00000060	e59b100c	46 LDR r1, [fp, #12] // print move disk n to to_rod
00000064	e59b2010	47 LDR r2, [fp, #16]
00000068	e59b0010	49 // do the same thing for (n-1, help_rod, to_rod)
0000006c	e52d08004	51 LDR r0, [fp, #16] // load to_rod
00000070	e51b0004	54 LDR r0, [r1, #16]
00000074	e52d08004	55 push {r0}
00000078	e59b0008	57 LDR r0, [fp, #8] // load in n-disk
0000007c	e2400001	58 SUB r0, r0, #1 // perform n - 1
00000080	e52d08004	59 push {r0}
00000084	ebfffffd	61 BL hanoi
00000088	e28dd00c	62 ADD sp, sp, #12 // Pop parameters
00000092	e59b100c	64 if_end:
00000096	00000000	66 // return
00000098	e59b2010	68 MOV sp, fp

Devices

- LEDs ff200000
- Switches ff200040
- Push buttons IRQ 73 ff200056
- Seven-segment displays ff200020
- JTAG UART IRQ 80 ff201000

Read FIFO: 0 Write FIFO: 0

Cortex-A9 Private Timer IRQ 29 fffec600
Once Stop TO=0

Cortex-A9 Watchdog Timer IRQ 30 fffec620
Once Stop TO=0 RST=0

HPS L4 Watchdog Timer IRQ 203 ff020000
2147483648 Interrupt Stop INTR=0

HPS L4 Watchdog Timer IRQ 204 ff030000
2147483648 Interrupt Stop INTR=0

Interval Timer IRQ 72 ff202000
6249999 Once Stop TO=0

From 2 to 3

Discord | #learn_tin | Teams and Channels | CS255 Syllabus | Grades for Ahmed A Adi | Tower Hanoi built - Go! | CPUlator ARMv7 System

cpulator.01xz.net/?sys=arm-de1soc

Stopped Step Into F2 Step Over Ctrl+F2 Step Out Shift+F2 Continue F3 Stop F4 Restart Ctrl+R Reload Ctrl+Shift+L File Help

Registers

r0	00000001
r1	00000003
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	fffffc34
sp	00000000
lr	fffffc30
pc	00000088
cpsr	600001d3 ZC I SVC
spsr	600001d3 ZC I SVC

Call stack **Trace** **Breakpoints** **Watchpoints** **Symbols** **Counters** **Settings**

Number Display Options
Size: Word Format: Hexadecimal Memory words per row: 4

Messages
Compiling...
Code and data loaded from ELF executable into memory. Total size is 184 bytes.
Assemble: arm-eabi-as -mfloat-abi=softfp -march=armv7-a -mpcu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmmTilkV.s.o work/asmmTilkV.s
Link: arm-eabi-ld --script build_arm.ld -e _start -u _start -u work/asmmTilkV.s.elf work/asmmTilkV.s
Compile succeeded.

</> Disassembly (Ctrl-D)

Go to address, label, or register: 00000000 Refresh

Address	Opcode	Disassembly
00000000	e52de004	4 PUSH {lr} // save return address and frame pointer
00000004	e52db004	5 PUSH {fp}
00000008	e1a0b00d	6 MOV fp, sp // make the base pointer
0000000c	e24dd004	7 SUB sp, sp, #4 // help rod variables
00000010	e59b0008	9 LDR r0, [fp, #8] // load n disks
00000014	e3500001	10 CMP r0, #1 // check if n == 0
00000018	1a000002	11 BNE e1else // else
0000001c	e59b100c	13 LDR r1, [fp, #12] // r1 = fromPeg
00000020	e59b2010	14 LDR r2, [fp, #16] // r2 = toPeg
00000024	ea000018	15 // we would print this
00000028	e59b000c	17 B if_end
00000032	00000000	19 else:
00000036	e59b2010	20 // call function for (n-1, from_rod, help_rod)
00000040	00000000	22 // Find help rod position
00000044	e59b100c	23 LDR r0, [fp, #12] // load from_rod
00000048	00000000	else:
00000052	e59b2010	1dr r0, [r1, #12]

Devices

- LEDs ff200000
- Switches ff200040
- Push buttons IRQ 73 ff200056
- Seven-segment displays ff200020
- JTAG UART IRQ 80 ff201000

Read FIFO: 0 Write FIFO: 0

Cortex-A9 Private Timer IRQ 29 fffec600
Once Stop TO=0

Cortex-A9 Watchdog Timer IRQ 30 fffec620
Once Stop TO=0 RST=0

HPS L4 Watchdog Timer IRQ 203 ff020000
2147483648 Interrupt Stop INTR=0

HPS L4 Watchdog Timer IRQ 204 ff030000
2147483648 Interrupt Stop INTR=0

Interval Timer IRQ 72 ff202000
6249999 Once Stop TO=0

From 1 to 3