

1. At the u-boot entry in DDR@0x80800000

TRACE32 ARM (POWER DEBUG USB @)

File Edit View Var Break Run CPU Misc Trace Perf Cgv TI-AM43xx AM43xx Linux-AM43xx Window Help

Registers/Spotlight

N	R0	40338DC4	R8	0
Z	R1	701FE019	R9	81FFFF20
C	R2	40337A04	R10	0
V	R3	80800000	R11	0
Q	R4	4032589c	R12	81FFF9EC
	R5	0	R13	81FFFEED8
0	R6	81FFFF00	R14	40302686
1	R7	403258FC	PC	80800000
2	SPSPR	0	CPSR	60000193
3				
4	USR:		FIQ:	
	R8	0	R8	0
I	R9	81FFFF20	R9	0
F	R10	0	R10	0
	R11	0	R11	0
J	R12	81FFF9EC	R12	0
T	R13	0	R13	0
	R14	0	R14	0
SV			SPSPR	0
nscc				

B:Var/Frame/Locals/Caller

t.u. Down Args Locals V C

-000|_image_copy_start(asm)

-001|NST:0x40302686(asm)

end of frame

B:DataList

Step Over Diverge Return Up Co Break Mode Find: vectors.S

addr/line code label mnemonic comment

```

_start:
#ifdef CONFIG_SYS_DV_NOR_BOOT_CFG
.word CONFIG_SYS_DV_NOR_BOOT_CFG
#endif

NSR:80800000 54 b reset 0x80800300 : reset
NSR:80800004 55 ldr pc, _undefined_instruction : pc, _undefined_instruction
NSR:80800008 56 ldr pc, _software_interrupt : pc, _software_interrupt
NSR:8080000C 57 ldr pc, _prefetch_abort : pc, _prefetch_abort
NSR:80800010 58 ldr pc, _data_abort : pc, _data_abort
NSR:80800014 59 ldr pc, _not_used : pc, _not_used
NSR:80800018 60 ldr pc, _irq : pc, _irq
NSR:8080001C 61 ldr pc, _fiq : pc, _fiq
NSR:80800020 62 _undefin.:dcd pc,0x80800038 : pc, _undefined_instruction
NSP:80800024 63 _softwar.:dcd 0x808000C0 : software_interrupt
NSP:80800028 64 _prefetc.:dcd 0x80800120 : prefetch_abort
NSP:80800032 65 _not_abu.:dcd 0x80800180 : data_abort
NSP:80800036 66 _not_used.:dcd 0x808001E0 : not_used
NSP:80800040 67 _irq: dcd 0x80800240 : irq
NSP:80800044 68 _fiq: dcd 0x808002A0 : fiq
NSP:80800048 69 DEADBEF dcd 0xDEADBEF : 
NSP:80800052 70 0BADCODE IRQ_STAC.:dcd 0x0BADCODE : 
NSR:80800056 71 320F000 nop : 
NSR:80800060 72 320F000 nop : 
NSR:80800064 73 320F000 nop : 
NSR:80800068 74 320F000 nop : 
NSR:80800072 75 320F000 nop : 
NSR:80800076 76 320F000 nop : 
NSR:80800080 77 320F000 nop : 
NSR:80800084 78 320F000 nop : 
NSR:80800088 79 320F000 nop : 
NSR:80800092 80 320F000 nop : 
NSR:80800096 81 320F000 nop : 
NSR:80800100 82 320F000 nop : 
NSR:80800104 83 320F000 nop : 
NSR:80800108 84 320F000 nop : 
NSR:80800112 85 320F000 nop : 
NSR:80800116 86 320F000 nop : 
NSR:80800120 87 320F000 nop : 
NSR:80800124 88 320F000 nop : 
NSR:80800128 89 320F000 nop : 
NSR:80800132 90 320F000 nop : 
NSR:80800136 91 320F000 nop : 
NSR:80800140 92 320F000 nop : 
NSR:80800144 93 320F000 nop : 
NSR:80800148 94 320F000 nop : 
NSR:80800152 95 320F000 nop : 
NSR:80800156 96 320F000 nop : 
NSR:80800160 97 320F000 nop : 
NSR:80800164 98 320F000 nop : 
NSR:80800168 99 320F000 nop : 
NSR:80800172 100 320F000 nop : 
NSR:80800176 101 320F000 nop : 
NSR:80800180 102 320F000 nop : 
NSR:80800184 103 320F000 nop : 
NSR:80800188 104 320F000 nop : 
NSR:80800192 105 320F000 nop : 
NSR:80800196 106 320F000 nop : 
NSR:80800200 107 320F000 nop : 
NSR:80800204 108 320F000 nop : 
NSR:80800208 109 320F000 nop : 
NSR:80800212 110 320F000 nop : 
NSR:80800216 111 320F000 nop : 
NSR:80800220 112 320F000 nop : 
NSR:80800224 113 320F000 nop : 
NSR:80800228 114 320F000 nop : 
NSR:80800232 115 320F000 nop : 
NSR:80800236 116 320F000 nop : 
NSR:80800240 117 320F000 nop : 
NSR:80800244 118 320F000 nop : 
NSR:80800248 119 320F000 nop : 
NSR:80800252 120 320F000 nop : 
NSR:80800256 121 320F000 nop : 
NSR:80800260 122 320F000 nop : 
NSR:80800264 123 320F000 nop : 
NSR:80800268 124 320F000 nop : 
NSR:80800272 125 320F000 nop : 
NSR:80800276 126 320F000 nop : 
NSR:80800280 127 320F000 nop : 
NSR:80800284 128 320F000 nop : 
NSR:80800288 129 320F000 nop : 
NSR:80800292 130 320F000 nop : 
NSR:80800296 131 320F000 nop : 
NSR:80800300 132 320F000 nop : 
NSR:80800304 133 320F000 nop : 
NSR:80800308 134 320F000 nop : 
NSR:80800312 135 320F000 nop : 
NSR:80800316 136 320F000 nop : 
NSR:80800320 137 320F000 nop : 
NSR:80800324 138 320F000 nop : 
NSR:80800328 139 320F000 nop : 
NSR:80800332 140 320F000 nop : 
NSR:80800336 141 320F000 nop : 
NSR:80800340 142 320F000 nop : 
NSR:80800344 143 320F000 nop : 
NSR:80800348 144 320F000 nop : 
NSR:80800352 145 320F000 nop : 
NSR:80800356 146 320F000 nop : 
NSR:80800360 147 320F000 nop : 
NSR:80800364 148 320F000 nop : 
NSR:80800368 149 320F000 nop : 
NSR:80800372 150 320F000 nop : 
NSR:80800376 151 320F000 nop : 
NSR:80800380 152 320F000 nop : 
NSR:80800384 153 320F000 nop : 
NSR:80800388 154 320F000 nop : 
NSR:80800392 155 320F000 nop : 
NSR:80800396 156 320F000 nop : 
NSR:80800400 157 320F000 nop : 
NSR:80800404 158 320F000 nop : 
NSR:80800408 159 3
```

2. At the entry of ENTRY(_main) in crt0.S (/arch/arm/lib/crt0.S)

TRACE32 ARM (POWER DEBUG USB @)

File Edit View Var Break Run CPU Misc Trace Perf Cgv TI-AM43xx AM43xx Linux-AM43xx Window Help

B:BreakList

B:DataList

addr/line	code	label	mnemonic	comment
NSR:80800620	76		#if defined(CONFIG_SPL_BUILD) && defined(CONFIG_SPL_STACK)	
			ldr sp, =(CONFIG_SPL_STACK)	
			#else	
			ldr sp, =(CONFIG_SYS_INIT_SP_ADDR)	
			#endif	
			#if defined(CONFIG_CPU_V7M)	/* v7M forbids using SP as BIC destination */
			mov r3, sp	
			bic r3, r3, #7	
			mov sp, r3	
			#else	
			bic sp, sp, #7	/* 8-byte alignment for ABI compliance */
			bic r13, r13, #0x7	; r13, r13, #
			mov r0, sp	
			cpy r0, r13	
			b1 board_init_f_alloc_reserve	
			b1x 0x8080BF7C	; board_init_f_alloc_reserve
			mov sp, r0	
			cpy r13, r0	
			/* set up gd here, outside any C code */	
			mov r9, r0	
			cpy r9, r0	
			b1 board_init_f_init_reserve	
			b1x 0x8080BF88	; board_init_f_init_reserve
			mov r0, #0	
			mov r0, #0x0	; r0, #0
			b1 board_init_f	
			b1x 0x8080DA5C	; board_init_f
			#if ! defined(CONFIG_SPL_BUILD)	

B:VarFrame/Locals/Caller

main(asm)

save_boot_params_ret(asm)

... /* the mask ROM code */

#ifndef CONFIG_SKIP_LOWLEVEL_INIT

b1 cpu_init_cp15

#ifndef CONFIG_SKIP_LOWLEVEL_INIT

b1 cpu_init_crit

#endif

#endif

b1 _main

components trace Data Var List PERF SYSTEM Step Go Break symbol Frame Register FPU other previous

NSR:80800620 \\u-boot\Global_main stopped MDX UP

In crt0.S (/arch/arm/lib/crt0.S) with a summary of u-boot boot flow

```
/*
 * This file handles the target-independent stages of the U-Boot
 * start-up where a C runtime environment is needed. Its entry point
 * is _main and is branched into from the target's start.S file.
 *
 * _main execution sequence is:
 *
 * 1. Set up initial environment for calling board_init_f().
 *    This environment only provides a stack and a place to store
 *    the GD ('global data') structure, both located in some readily
 *    available RAM (SRAM, locked cache...). In this context, VARIABLE
 *    global data, initialized or not (BSS), are UNAVAILABLE; only
 *    CONSTANT initialized data are available. GD should be zeroed
 *    before board_init_f() is called.
 *
 * 2. Call board_init_f(). This function prepares the hardware for
 *    execution from system RAM (DRAM, DDR...). As system RAM may not
 *    be available yet, board_init_f() must use the current GD to
 *    store any data which must be passed on to later stages. These
 *    data include the relocation destination, the future stack, and
 *    the future GD location.
 *
 * 3. Set up intermediate environment where the stack and GD are the
 *    ones allocated by board_init_f() in system RAM, but BSS and
 *    initialized non-const data are still not available.
 *
 * 4a. For U-Boot proper (not SPL), call relocate_code(). This function
 *    relocates U-Boot from its current location into the relocation
 *    destination computed by board_init_f().
 *
 * 4b. For SPL, board_init_f() just returns (to crt0). There is no
 *    code relocation in SPL.
 *
 * 5. Set up final environment for calling board_init_r(). This
 *    environment has BSS (initialized to 0), initialized non-const
 *    data (initialized to their intended value), and stack in system
 *    RAM (for SPL moving the stack and GD into RAM is optional - see
 *    CONFIG_SPL_STACK_R). GD has retained values set by board_init_f().
 *
 * 6. For U-Boot proper (not SPL), some CPUs have some work left to do
 *    at this point regarding memory, so call c_runtime_cpu_setup.
 *
 * 7. Branch to board_init_r().
 *
 * For more information see 'Board Initialisation Flow' in README.
 */
```

Note that:

- No symbol relocation is needed when JTAG debugging board_init_f() (/common/board_f.c)
- Symbol relocation is necessary when JTAG debugging board_init_r() (/common/board_r.c)
- Symbol relocation offset is listed in serial output when #define DEBUG is added in "/common/board_f.c".
- An alternative to get relocation offset is listed in next pages, where detail on relocation switching is shown.
- The u-boot banner (the very first line below), and rest of serial output listed below is from board_init_f()

U-Boot 2019.01-ge219a8dfce-dirty (Sep 09 2020 - 05:31:51 -0500)

U-Boot code: 80800000 -> 8086CF10 BSS: -> 8089FFD8

CPU : AM437X-GP rev 1.1

Model: TI AM437x GP EVM

DRAM: Monitor len: 0009FFD8

Ram size: 80000000

Ram top: 00000000

TLB table from ffff0000 to ffff4000

Reserving 639k for U-Boot at: fff50000

Reserving 32832k for malloc() at: fdf40000

Reserving 104 Bytes for Board Info at: fdf3ff98

Reserving 224 Bytes for Global Data at: fdf3feb8

Reserving 62528 Bytes for FDT at: fdf30a78

show_dram_config: 209:

RAM Configuration:

Bank #0: 80000000 2 GiB

Bank #1: 0 0 Bytes

Bank #2: 0 0 Bytes

Bank #3: 0 0 Bytes

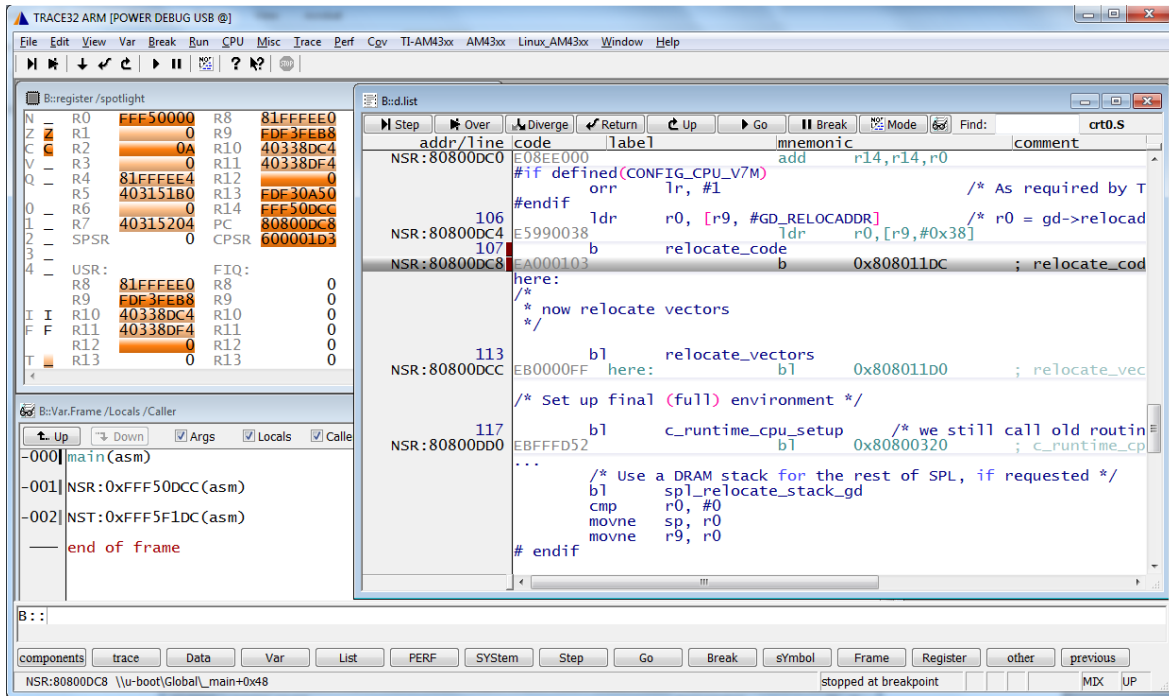
DRAM: 2 GiB

New Stack Pointer is: fdf30a50

Relocation Offset is: 7f750000

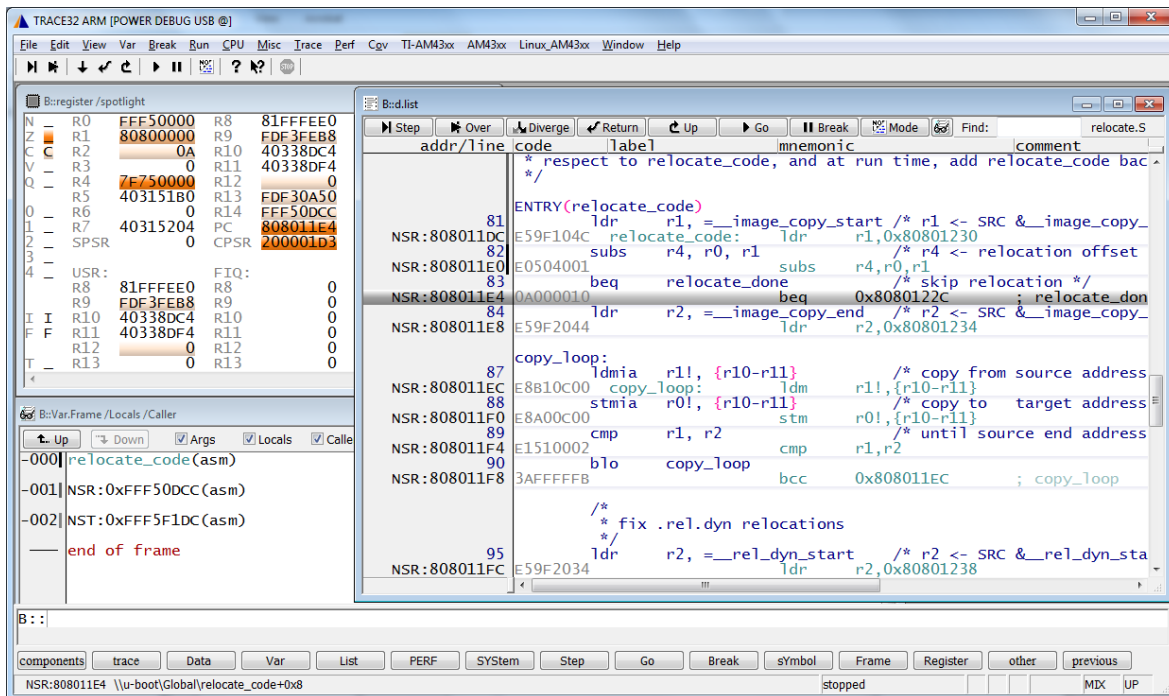
Relocating to fff50000, new gd at fdf3feb8, sp at fdf30a50

3. Set bkpt @relocate_code:



4. Step into relocate_code.

The relocation offset is in R4=**0x7F750000** in this example



5. At the exit of `relocate_code`, this is the last instruction of u-boot w/o relocation.

The screenshot shows the TRACE32 ARM debugger interface. The left pane displays the register/spotlight window with registers R0 through R13 and SPSR, CPSR. The right pane shows the disassembly window with the following code:

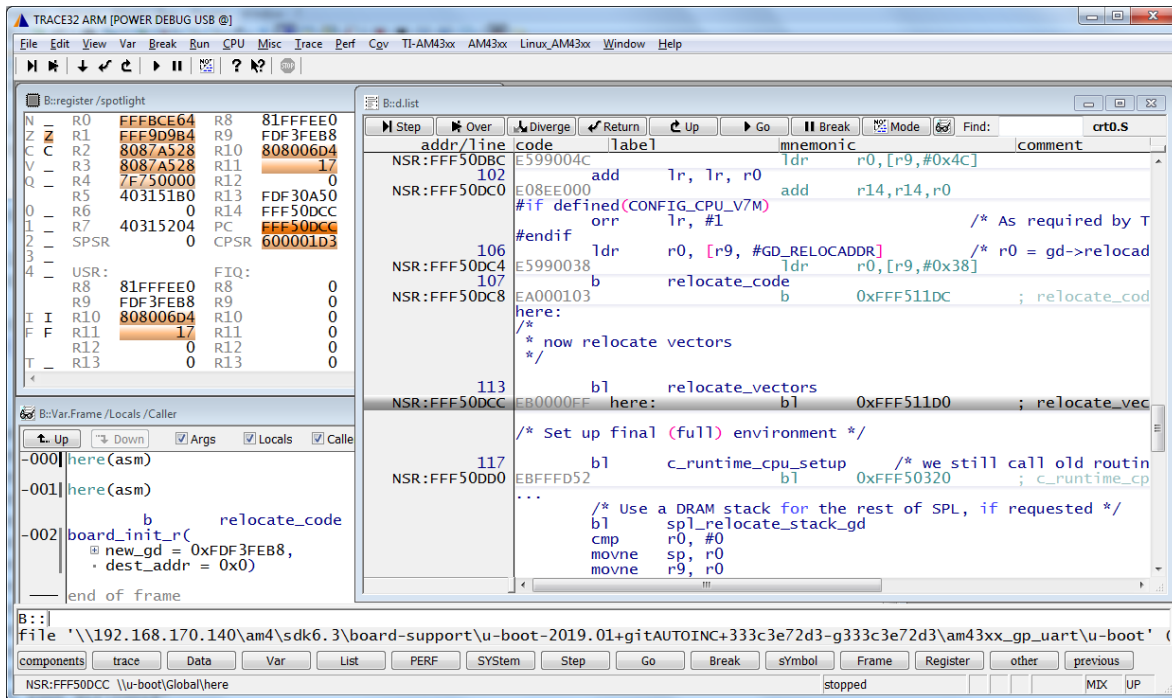
```
NSR:80801220  E5801000  str    r1,[r0]
fixnext:
NSR:80801224  E1520003  cmp    r2, r3
fixloop:
NSR:80801228  3AFFFFF5  bcc    0x80801204 ; fixloop
...
mcr    p15, 0, r0, c7, c10, 4 /* drain write buffer */
#endif
/* ARMv4- don't know bx lr but the assembler fails to see t
#ifdef __ARM_ARCH_4__
mov    pc, lr
#else
bx     lr
#endif
relocate_done: bx     r14
...
NOTE:
* To prevent the code below from containing references with an R_A
* relocation record type, we never refer to linker-defined symbols
* Instead, we declare literals which contain their relative locati
* respect to relocate_code, and at run time, add relocate_code bac
*/
ENTRY(relocate_code)
ldr    r1, __image_copy_start /* r1 <- SRC & __image_copy_
dcd    0x80800000 ; __image_copy_
```

6. One step further, u-boot after relocation

The screenshot shows the TRACE32 ARM debugger interface. The left pane displays the register/spotlight window with registers R0 through R13 and SPSR, CPSR. The right pane shows the disassembly window with the following code:

```
NSR:FFF50B88  E28FE00C  adr    r14, 0xFFFF50DCC
ldr    r0,[r9, #0x4C]
NSR:FFF50B8C  E599004C  ldr    r14, r14, r0
NSR:FFF50B90  E08EE000  add    r0,[r9, #0x38]
NSR:FFF50B94  E5990038  ldr    r0,[r9, #0x38]
NSR:FFF50B98  EA000103  b      0xFFFF511DC
NSR:FFF50B9C  FB0000FF  bl     0xFFFF511D0
NSR:FFF50BA0  EBFFFD52  bl     0xFFFF50320
NSR:FFF50BA4  E59F0028  ldr    r0, 0xFFFF50E04
NSR:FFF50BA8  E59F3028  ldr    r3, 0xFFFF50E08
NSR:FFF50BAB  E3A01000  mov    r1, #0x0 ; r1, #0
NSR:FFF50BAC  E0532000  subs   r2, r3, r0
NSR:FFF50BAD  FA00000D  blx    0xFFFF50E20
NSR:FFF50BAE  FB00383C  blx    0xFFFF5EEE2
NSR:FFF50BAB  FA00383C  blx    0xFFFF5EEE4
NSR:FFF50BFC  E1A00009  cpy    r0, r9
NSR:FFF50BFD  E5991038  ldr    r1, [r9, #0x38]
NSR:FFF50BFE  E59FE00C  ldr    r14, 0xFFFF50E0C
NSR:FFF50BF0  E12FFF1E  bx     r14
NSR:FFF50BF4  4033FF20  eorsmi pc, r3, r0, lsr #0x1E
NSR:FFF50BF8  FFFBCF10  svc    0xFBCF10
NSR:FFF50BF8  FFFFEFD8  svc    0xFFFFFD8
NSR:FFF50BF8  FFF5F1D0  svc    0xF5F1D0
NSR:FFF50BF8  00000000  andeq  r0, r0, r0
NSR:FFF50BF8  00000000  andeq  r0, r0, r0
NSR:FFF50BF8  00000000  andeq  r0, r0, r0
NSR:FFF50BF8  00000000  andeq  r0, r0, r0
NSR:FFF50BF8  0303F010  movewq pc, #0x3010
NSR:FFF50E24  D13D4684  teqle  r13, r4, lsr #0x0D
```

- Relocate the u-boot symbol using <y.reloc 0x7F750000> for T32 JTAG debugger. The screenshot in #6 is shown below which matches u-boot SRC.



- The serial output listed below is from board_init_r() (/common/board_r.c)
- Some of serial output is due to "#define DEBUG" added in (/common/board_r.c) as an example.
- U-boot stops at u-boot prompt now...

Pre-reloc malloc() used 0xb50 bytes (2 KB)
 Now running in RAM - U-Boot at: fff50000
 PMIC: TPS65218
 NAND: 512 MiB
 MMC: OMAPSD/MMC:0
 Loading Environment from FAT...

Net:
 Warning: ethernet@4a100000 using MAC address from ROM
 eth0: ethernet@4a100000
 Hit any key to stop autoboot: 0
 =>