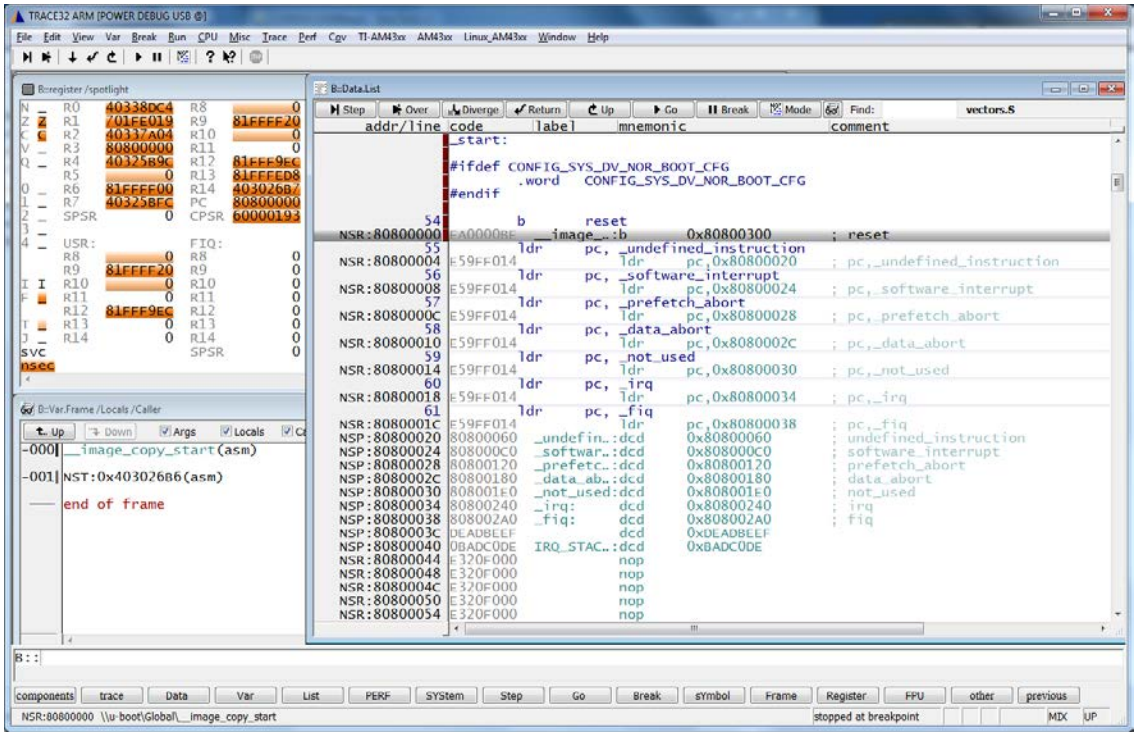
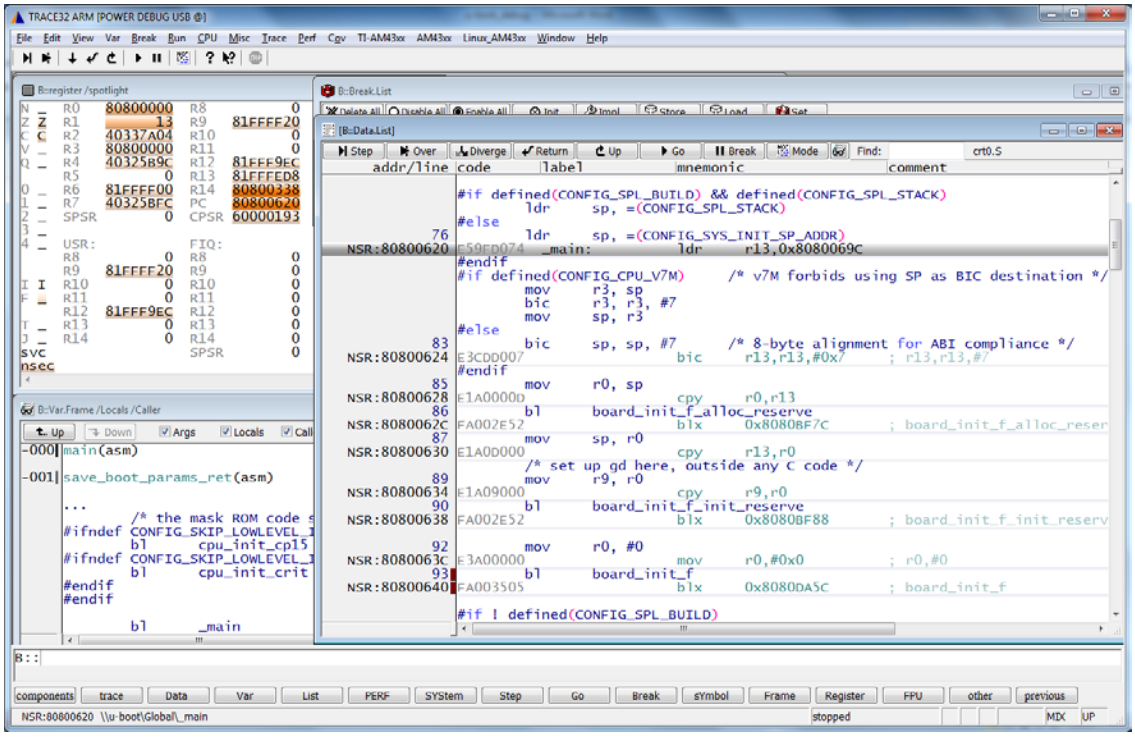
u-boot boot flow (AM437x GP is used as an example)

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



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



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()

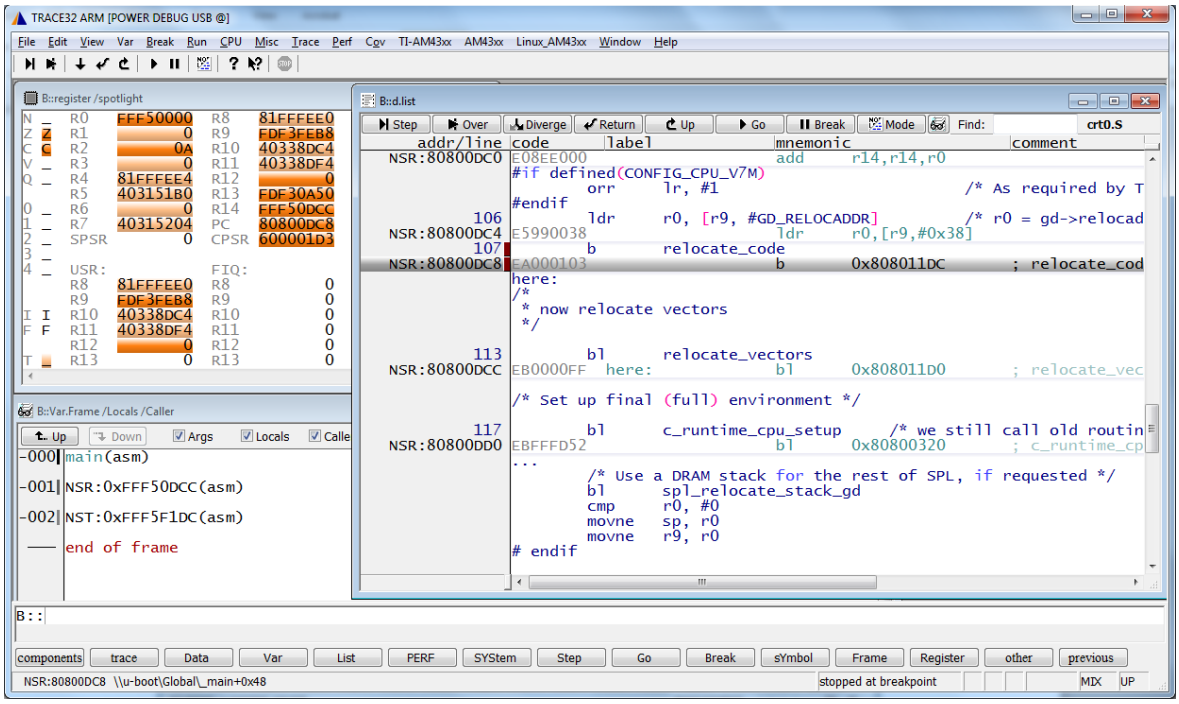
U-Boot 2019.01-ge219a8dfce-dirty (Sep 09 2020 - 05:31:51 -0500)

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:

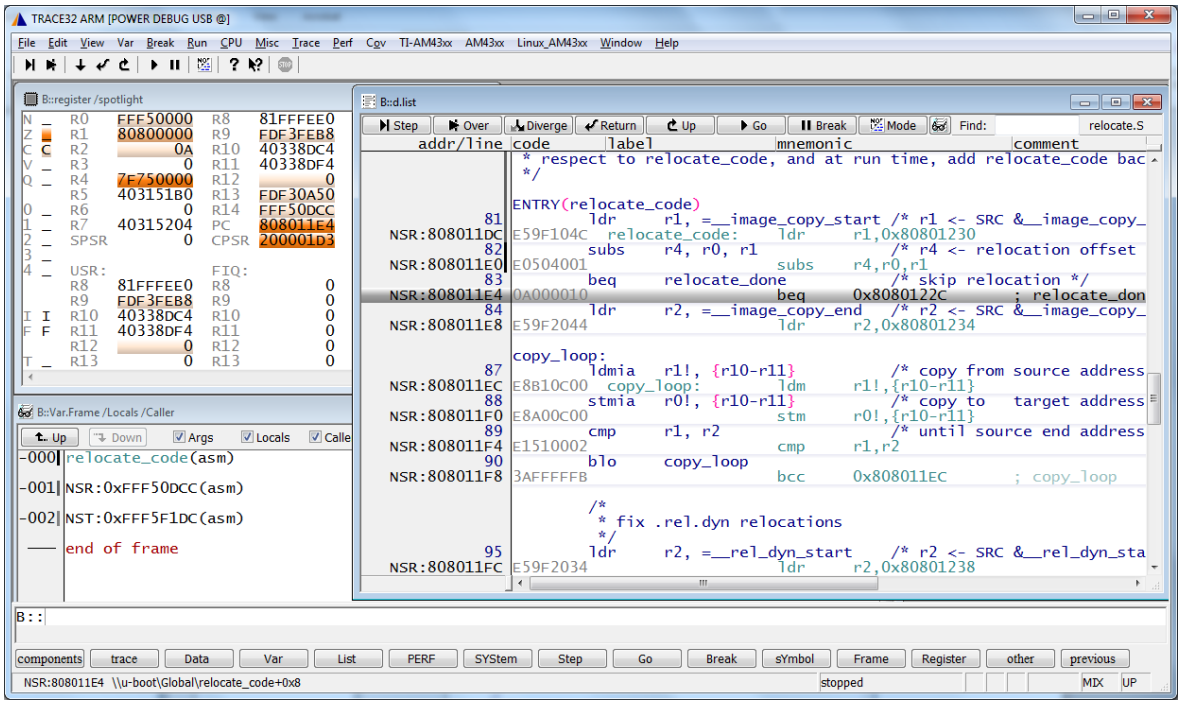
RAM Configuration:
Bank #0: 80000000 2 GiB
Bank #1: 0 0 Bytes
Bank #2: 0 0 Bytes
Bank #3: 0 0 Bytes

DRAM: 2 GiB
New Stack Pointer is: fdf30a50
Relocation Offset is: 7f750000
Relocating to fff50000, new gd at fdf3feb8, sp at fdf30a50

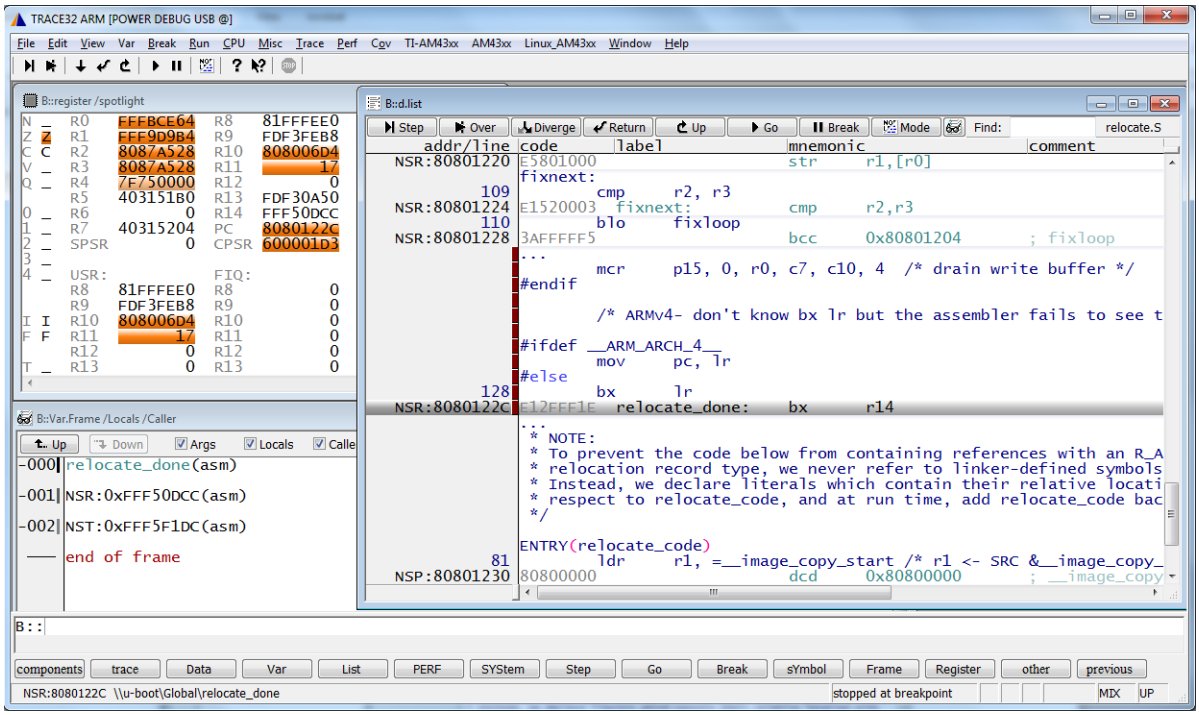
3. Set bkpt @relocate_code:



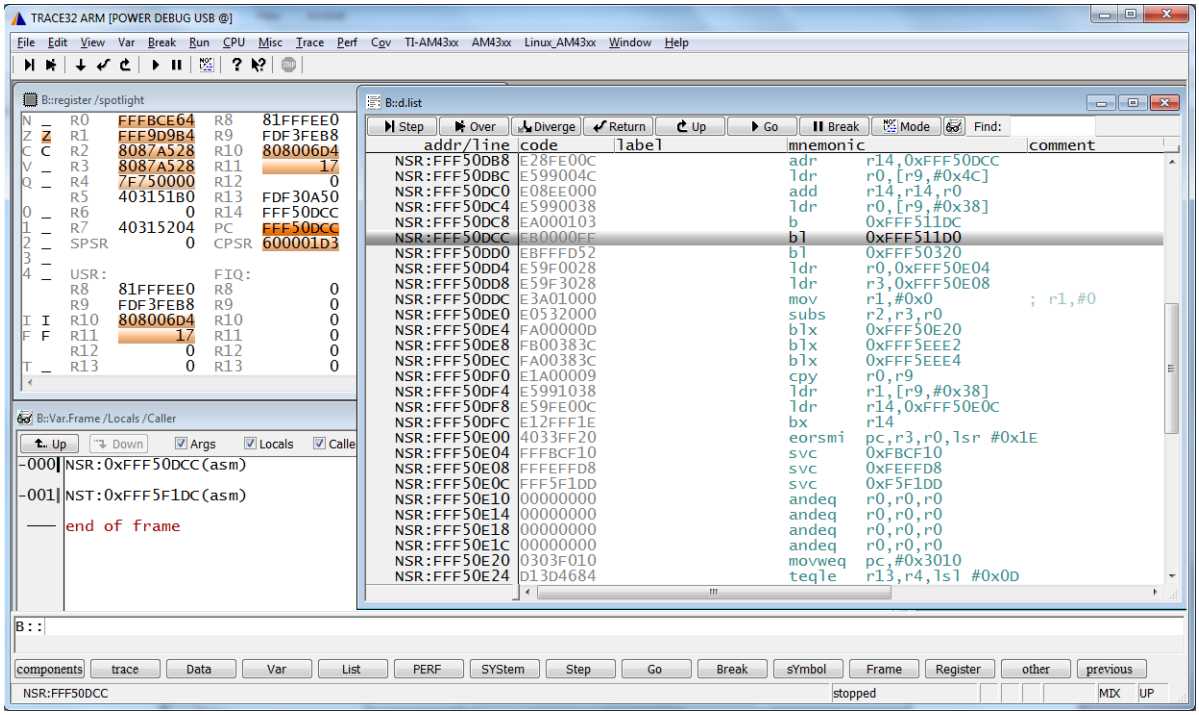
4. Step into relocate_code.
   The relocation offset is in R4=0x7F750000 in this example



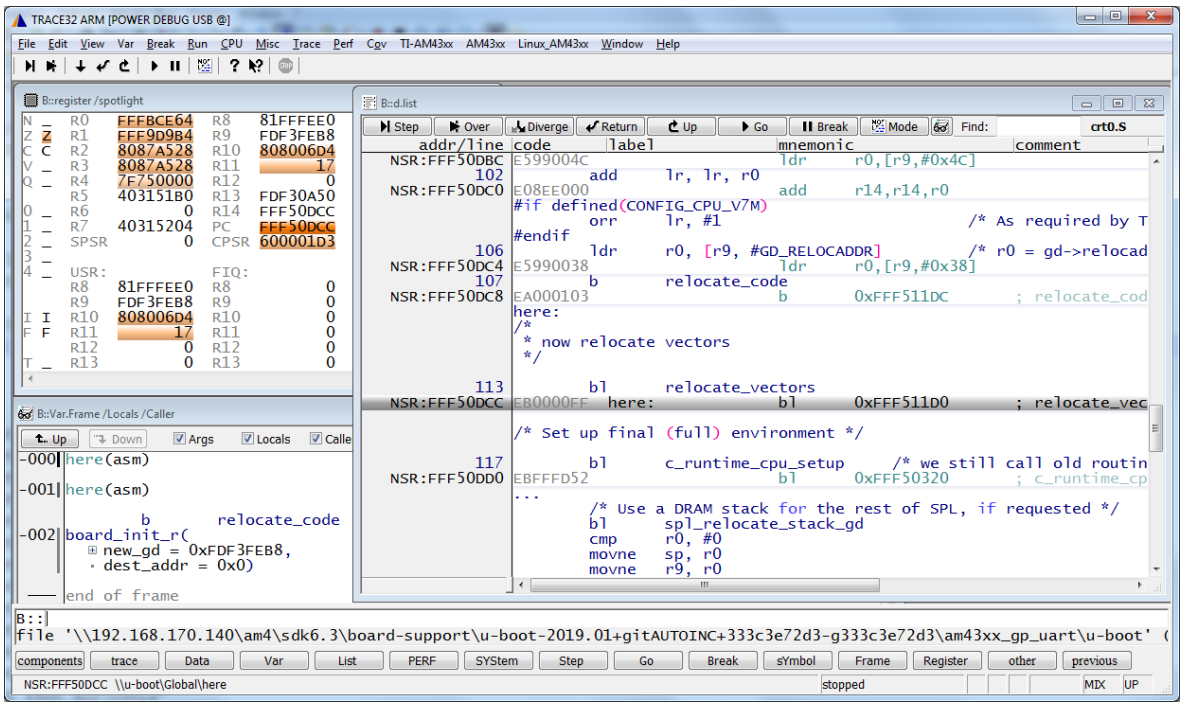
5. At the exit of relocate_code, this is the last instruction of u-boot w/o relocation.



6. One step further, u-boot after relocation



7. 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:   OMAP SD/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
=>
```