This document aims to testing correctness of ARM-instruction Decoding functions

Test approach

Down to top

1)down: test low-level individual decoding functions

Using GDB expression evaluation or using internal test-functions

Low-level Functions to be tested

1. CPU\_INT08U Debug\_HAL\_INST\_Is\_Condition\_True(CPU\_INT32U Instruction,Debug\_TID\_t ThreadID)

Given that :

Thread ID is 8

CPSR at task 8 is ??

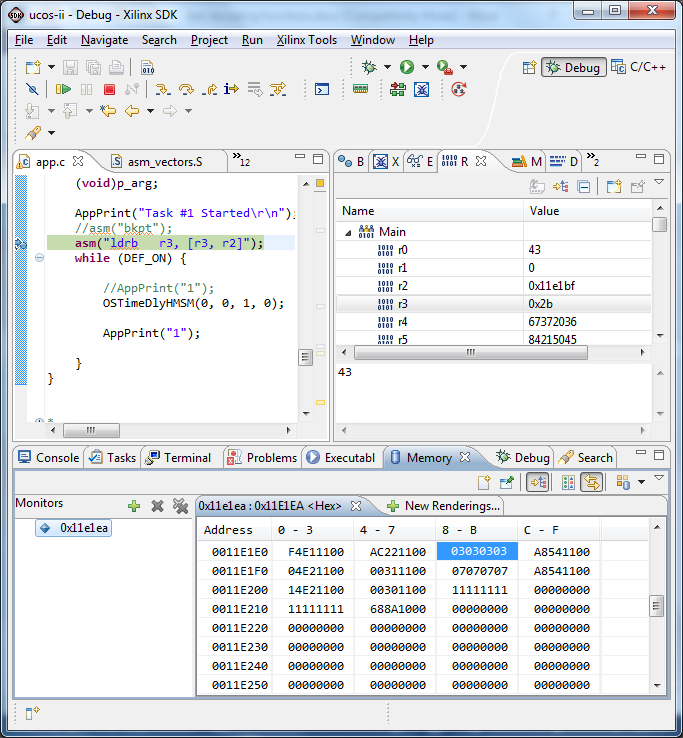
|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | ThreadID | Expected flag | Real flag |
| 0x012FFF12 | 8 |  |  |
| 0x112FFF12 |  |  |  |
| 0x212FFF12 |  |  |  |
| 0x312FFF12 |  |  |  |
| 0x412FFF12 |  |  |  |
| 0x512FFF12 |  |  |  |
| 0x612FFF12 |  |  |  |
| 0x712FFF12 |  |  |  |
| 0x812FFF12 |  |  |  |
| 0x912FFF12 |  |  |  |
| 0xA12FFF12 |  |  |  |
| 0xB12FFF12 |  |  |  |
| 0xC12FFF12 |  |  |  |
| 0xD12FFF12 |  |  |  |
| 0xE12FFF12 |  |  | 1 |
| 0xF12FFF12 |  |  | 223 🡪 unconditioned |

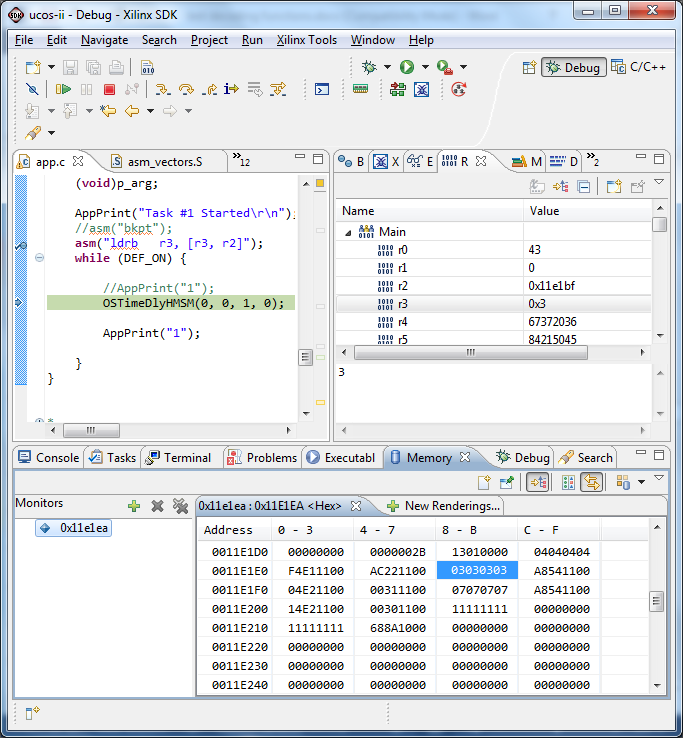
1. static Debug\_MemWidth Get\_Target\_DP\_Class(CPU\_INT32U Instruction)

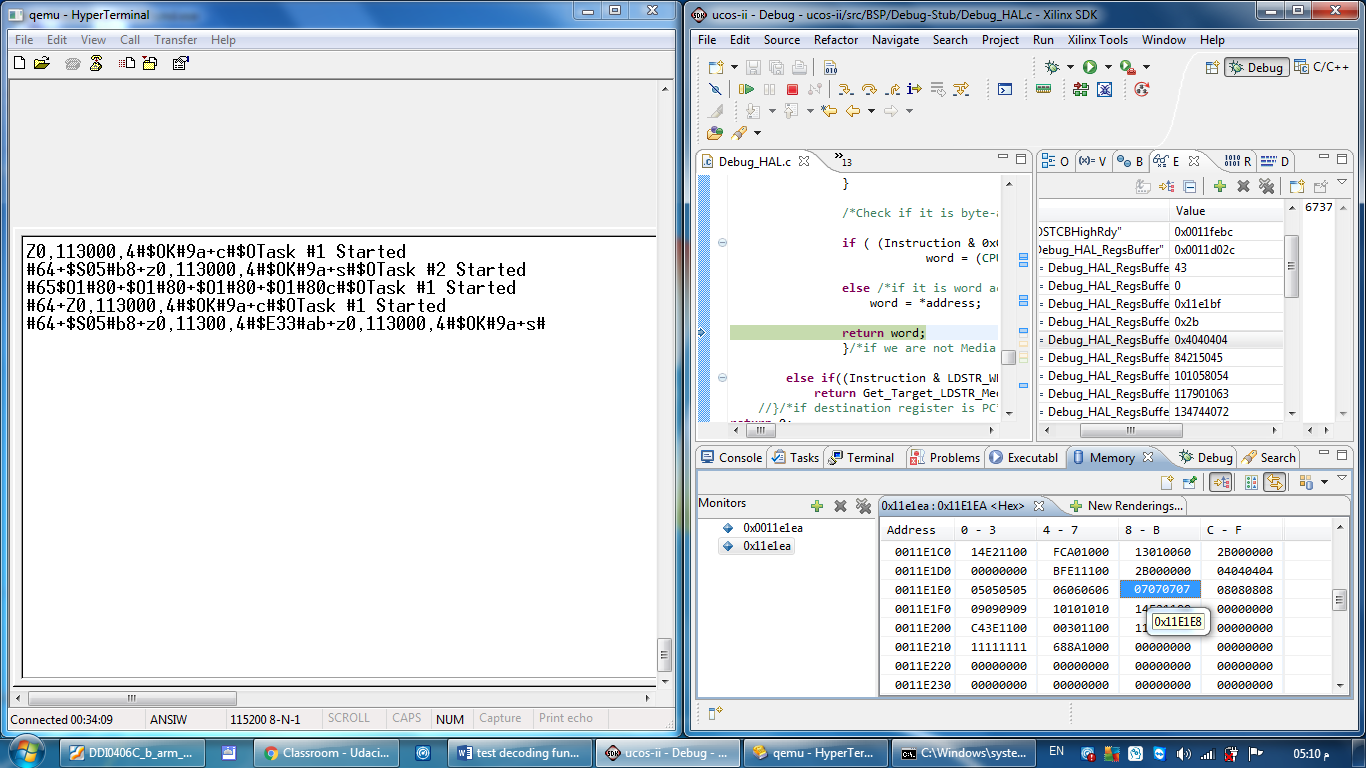
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | | Test Address | Thread id | | Expected Resuly | Real result | Injected or on elf |
| "BX Rm" 0x<Cond>12FFF1<Rm> | | | | | | |  |
| 0xE12FFF12 | |  |  | |  |  |  |
| 0xE12FFF14 | |  |  | |  |  |  |
| 0xE12FFF1F  Rm is PC | |  |  | |  |  |  |
| : bx lr  0xe12fff1e | | 0x108af4  End of OS\_TaskStkClr | 8🡪 starter task | | LR=0x107c14  And it returns to to it | 0x107c14 | elf |
| Bx r0  0xe12fff10 | |  |  | |  |  |  |
|  | "BLX Rm" 0x<Cond>12FFF6<Rm> | | | | | |  |
| 0xe12FFF32 | |  |  | |  |  |  |
| 0xe12FFF34 | |  |  | |  |  |  |
| 0xe12FFF3F | |  |  | |  |  |  |
| 0xe12fff33  blx r3 | |  |  | | R3 at this time is | right | injected |
|  | |  | | "ERET" 0x<Cond>160006E | | |  |
|  | |  |  | |  |  |  |
| 0xe160006E : ERET | | None | 8 | | 0x101f60 | [LR] = 0x101f60 | injected |
|  | |  | | "BKPT #immediate" 0x<Cond>1200070 | | |  |
| 0xe1200070  Bkpt 0 | | Need special handling for decoding and stepping |  | |  | 0x00100000 + c = 0x0010000C | Inject asm("bkpt") |
| Registers,shift registers,immediate | | | | | | | |
| 102390 : e2033007  AND r3,r3,#7 | |  |  | |  |  | Elf-in-wa-y |
| 10f438: e0223003 eor r3, r2, r3  EOR | |  |  | |  |  | Elf-not in-wa-y |
| 112af4: e28db004  ADD fp,sp,#4 | |  |  | |  |  | Elf-in-wa-y |
| 11485c: e0a22002 adc r2, r2, r2  ADC | |  |  | |  |  | Elf-not in-way |
| 112fa8: e24dd008  SUB sp,sp,#8 | |  |  | |  |  | Elf-in-wa-y |
| 10e76c: e2e31000 rsc r1, r3, #0  RSC | |  |  | |  |  | Elf-I not n-wa-y |
| 114888 : e262201f  RSB r2, r2, #31 | |  |  | |  |  | Elf-not in way"injected" |
| RRX | |  |  | |  |  | injected |
| SBC | |  |  | |  |  | Injected |
| 104ad4 :e38330ff  ORR r3,r3,#255 | |  |  | |  |  | Elf-in-wa-y |
| 100174: e3c00001 bic r0, r0, #1  BIC | |  |  | |  |  | Elf- not in-way |
| 10fb60: e1a03372 ror r3, r2, r3  ROR | |  |  | |  |  | Elf-in-wa-y |
| 1016f0: e1e03003 mvn r3, r3  MVN | |  |  | |  |  | Elf- not in-way |
| ADR | |  |  | |  |  | injected |
| 104da4 : e1a03000  MOV r3,r0 | |  |  | |  |  | Elf-in-wa-y |
| 104ecc : e1a03403  LSL r3,r3,#8 | |  |  | |  |  | Elf-in-wa-y |
| 10237c : e1a031a3  LSR r3,r3,#3 | |  |  | |  |  | Elf-in-wa-y |
| 104dc4 : e3530000  CMP r3,#0 | |  |  | |  |  | Elf-in-wa-y |
| 109f7c: e31e0001 tst lr, #1  TST | |  |  | |  |  | Elf-not in way |
| TEQ | |  |  | |  |  | injected |
| 10fe54: e3730001 cmn r3, #1  CMN | |  |  | |  |  | Elf- not in way |
| 100390: e1a03143 asr r3, r3, #2  ASR | |  |  | |  |  | Elf-not in-way |
| Misc | | | | | | | |
|  | |  |  | |  |  |  |
|  | |  |  | |  |  |  |
|  | |  |  | |  |  |  |

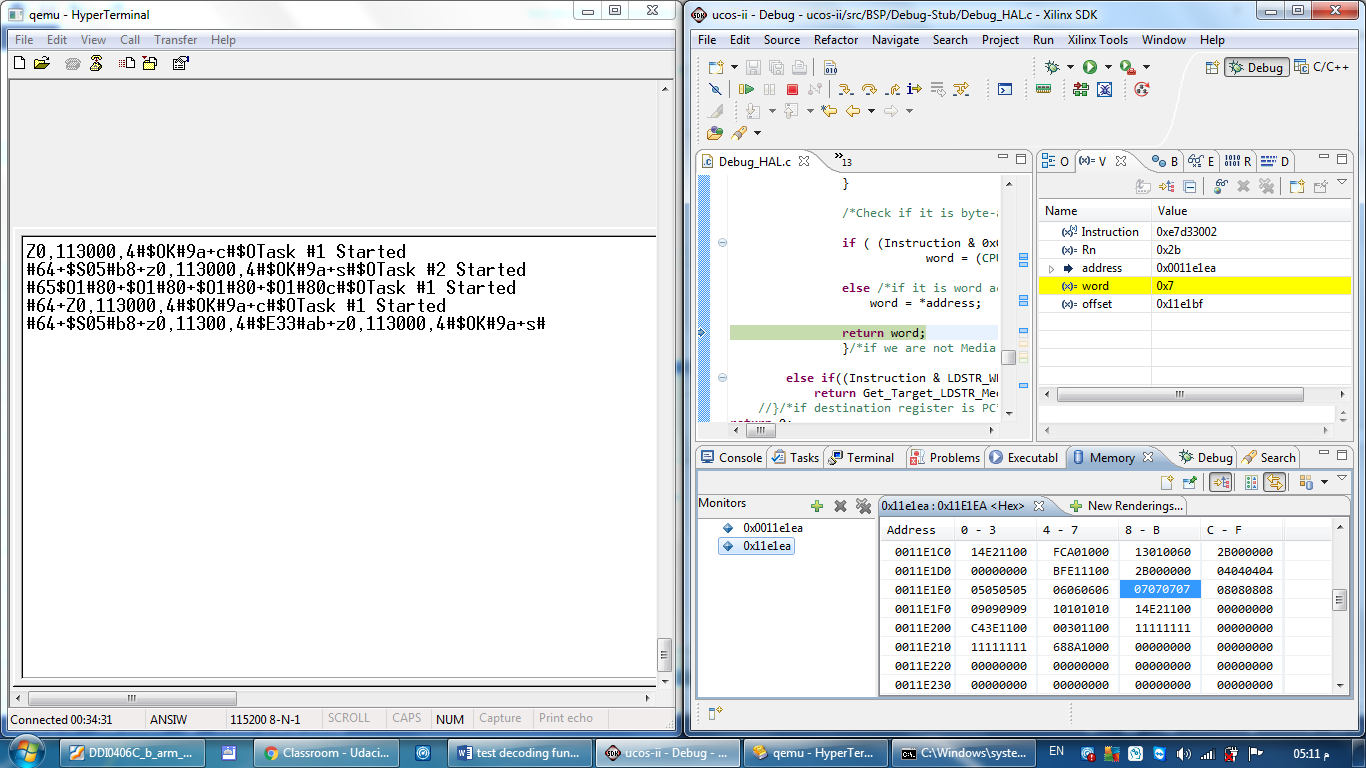
1. static Debug\_MemWidth Get\_Target\_LDSTR\_Class(CPU\_INT32U Instruction)

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Real | Expected | Elf or injected |
| 111480:e5d33001  ldrb r3, [r3, #1] |  |  |  |
| 107bc4: e5d33000 ldrb r3, [r3] | 0 | R3 = 0 | Elf,inway |
| 107c0c: e51b0008 ldr r0, [fp, #-8] | target 0x80000113  After code correction in  "address = Rn - ( ((Instruction & 0x01000000)>> 24)/\*if P is set, add offset\*/ \*offset ); "  Required code correction | R0 = 0x80000113 | Elf,inway |
| 10000c: Ldr pc,[pc,#20] e59ff014 |  |  | Elf |
| 107bec: e7933102 ldr r3, [r3, r2, lsl #2] | Target = 0 | R3 =0 | Elf,inway |
| 107bf8: e55b2021 ldrb r2, [fp, #-33] ; 0xffffffdf | Target 2 | R2 = 2 | Elf,inway |
| 113000: e7d33002 ldrb r3, [r3, r2] | Target = 7  This relates to task stack context is untuned asin below screenshot  I do not expect this as error . it just need to be put in its real context | R3 = 3 | Code injection (asm) in AppTask1  Elf,not in way |
| 112fd8: e5832000 str r2, [r3]  Str | 0 | 0 | Elf,inway |
| 104e38: e7831102 str r1, [r3, r2, lsl #2]  Str | 0 | 0 | Elf,in way |
| 104dac: e54b300d strb r3, [fp, #-13]  Strb | 0 | 0 | Elf,in way |
| 104ec0: e5c32000 strb r2, [r3] | 0 | 0 | Elf,in way |
| 102594: e7c31002 strb r1, [r3, r2]  Strb | 0 | 0 | elf,inway |









1. static Debug\_MemWidth Get\_Target\_Branch\_Class(CPU\_INT32U Instruction)

|  |  |  |  |
| --- | --- | --- | --- |
| Passed Instruction | Real | Expected | In elf or injected |
| "B label" 0x<cond>A<imm24> | | |  |
| 0xeA123456 |  |  |  |
| 0xeAF23456 |  |  |  |
| 0xeAE23455 |  |  |  |
| 108c10: ea000002 b 108c20 <OSTimeDly+0x128> | 108c20 | 108c20 |  |
| "BL(X) <Label>" 0x<Cond>B | | |  |
| 0xeb123456 |  |  |  |
| 0xebF23457 |  |  |  |
| AppMutexCreate(); 112da4:eb000043 bl 112ec0 <AppMutexCreate> |  | 00112eb8 | Elf |
| ebfffe8e  111904 : bl 111344 <Debug\_RSP\_Put\_Packet> |  | 111344 |  |
| LOAD/STORE | | |  |
| LOAD | | |  |
| 107ce0: e8bd8800 pop {fp, pc}  End of OSTaskCreateExt() | Target\_Address = 0x00112f58 | 112f58 :e3a03002 mov r3, #2) | Elf,in way |
| 109fd0: e8fddfff ldm sp!, {r0, r1, r2, r3, r4, r5, r6, r7, r8, r9, sl, fp, ip, lr, pc}^ | 101f60 | 00101f60: ldr r0, [r11, #-8] | Elf , in way |
| 0xe89 F 8101  Load inc after Rn=F , PC in reglist |  |  |  |
| 0xe89 D 8101 🡪 POP  Load inc after Rn= !F Rn = SP , PC in reglist |  |  |  |
| 0xe89 F 6101  Load inc after PC not in reglist |  | 0 |  |
| 11305c : e8ac000f stmia ip!, {r0, r1, r2, r3} | 0 | 0 | Elf,in way |
| 0xe99 F 8101  Load inc before Rn=F , PC in reglist |  |  |  |
| 0xe99 6 8101  Load inc before Rn= !F , PC in reglist |  |  |  |
| 0xe99 F 6101  Load inc before PC not in reglist |  | 0 |  |
| 0xe81 F 8101  Load decrement after Rn=F , PC in Reglist |  |  |  |
| 0xe81 6 8101  Load decrement after Rn !=F , PC in Reglist |  |  |  |
| 0xe81 6 7101  Load decrement after PC not in Reglist |  | 0 |  |
| 0xe91 F 8101  Load decrement before Rn=F , PC in Reglist |  |  |  |
| 0xe91 6 8101  Load decrement before Rn !=F , PC in Reglist |  |  |  |
| 0xe91 6 7101  Load decrement before PC not in Reglist |  | 0 |  |
| STORE | | |  |
| 0xe88 5 F000  Store Increment after |  | 0 |  |
| 0xe98 5 8000  Store Increment before | 0 | 0 | Injected |
| 0xe80 F F000  Store Decrement after | 0 | 0 | injected |
| 0xe90 F 8000  Store Decrement before |  | 0 |  |
| SVCHandler: /\* SWI handler \*/  stmdb sp!,{r0-r3,r12,lr} /\* state save from compiled code \*/  100068: e92d500f push {r0, r1, r2, r3, ip, lr} | 0 | 0 | In elf, injected |
| stmdb sp!,{r0-r3,r12,lr}  10008c: e92d500f push {r0, r1, r2, r3, ip, lr} |  | 0 |  |
| 0x8D 5 8101  Load exception return  Inc = 1 , wordhigher = 0 Rn=R5 PC in reglist |  |  |  |
| 0x9D 5 8101  Load exception return  Inc = 1 , wordhigher = 1 Rn=R5 PC in reglist |  |  |  |
| 0x85 5 8101  Load exception return  Inc = 0 , wordhigher = 1 Rn=R5 PC in reglist |  |  |  |
| 0x85 5 6101  Load user register  Inc = 0 , wordhigher = 1 Rn=R5 PC not in reglist |  | 0 |  |

1. static Debug\_MemWidth Get\_Target\_COPSVC\_Class(CPU\_INT32U Instruction)

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Real | Expected | Elf |
| 0xeF000000  113060 ef000001 svc 0x00000001  We can not debug SVC handler as it loops in infinite loop between prefetch handler and OS\_CPU\_ARM\_ExceptHndlr which already has a stub breakpoint  Solution , put a breakpoint after the SVC | Target = 100008  Then  0x10a024 | 0x00100000 + 8 = 0x00100008 | In elf, code injection  In AppTask1 : asm("SVC #1");  Subsequent <s> packets caused a Data\_Abort Exception to occur caused by during prefetchabort handler in handling bkpt at 0x10a070  10a038: e92d5fff push {r0, r1, r2, r3, r4, r5, r6, r7, r8, r9, sl, fp, ip, lr} |
| Undefined  What would we return from Get\_Target\_COPSVC\_Class()   1. 0 2. Special flag 3. Or just 0x00100004 | No  Stub sends S04 to indicated SigILL  if GDB then sends <s> the pc to step from is the one after undefined exception | 0x00100004 | 1. Via injection (**Get\_Target\_COPSVC\_Class(**0xeC000000**)**) 2. Via <M>packet with undefined opcode in AppTask1 code area:    1. M113068#    2. M113068,eC000000 3. Via asm(b <data area>)   In handling undefined exception processor sets pc at next instruction after undefined-excep location  Thus ,ucos-ii not decrementing LR  Stub will not be able to step into exception it handles in its gdb\_exception handler due to   1. Task stack is not correct (as context is not saved) 2. Stub will not wakeup due to nested exception occurance. |
| Data abort handler | Discuss if we need to advance pc by 4 to skip the exception-point address |  |  |
|  |  |  |  |
|  |  |  |  |

1. static Debug\_MemWidth Get\_Target\_LDSTR\_Media(CPU\_INT32U Instruction);
2. static CPU\_INT32U Shift\_Shift\_C(CPU\_INT32U operand, CPU\_INT08U shift\_t, CPU\_INT08U shift\_n);
3. static CPU\_INT08U Count\_NumRegsLoaded( CPU\_INT16U CPU\_RegList);

Higher level functions to be tested

1. Debug\_HAL\_INST\_Get\_Target\_Address
   1. Does it enter write case
   2. Does it call right functions with right parameters
2. CPU\_INT08U Debug\_Main\_Step\_machine\_instruction(Debug\_TID\_t ThreadID,void \*CommandParam)

Test Data

Test data would be sample instructions to test how functions decode them and extract out the target PC for next instruction.

Test-instruction requires address-machine\_code-assembly combination to test

Address : where we would step from

Assembly: assembly instruction that would extract instruction class and parameters from it

Machine code: the binary code of instructions that we would use to test whether decoding functions enter right paths

All these information are combined in one line in ucos-ii.elf file when opened in Xilinx-SDK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address | Assembly | Machine code(Encoded instruction) | Expected target PC | Calculated target PC | Correct? |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Calculating Expected Result

Expected result could be obtained from debugging ucos-ii through Qemu's Stub

For example :

In .elf file task2 function

AppPrint("Task #2 Started\r\n");

112e5c: e30502f4 movw r0, #21236 ; 0x52f4

112e60: e3400011 movt r0, #17

112e64: eb00001e bl 112ee4 <AppPrint>

We need to know what real target address of bl instruction at address 0x112e64

It could be known in two ways

Easy way : get address of AppPrint in .elf file and this is the expected result of decoding PC instruciton at address 0x112e64

Second way is to use gdb with Wemustub to step single instruction from address 0x112e64

GDB> si 0x112e64

GDB > print $pc

Now gdb would display line it stopped in ,we will map this line to its corresponding address in elf file and this would be result of encoding instruction at address 0x112e64

Or us PC register value

Issues regarding the thread context assume current implementation

1. if using step address in <s> packet,
   1. when decoding and fetching the register values as needed in instruction operands itself , we are getting the current thread context not the new expected context.
   2. Second <s> packet would not decode next instruction but it would redecode same instruction as its previous <s> packet

Issues encountered during testing following instruction

1. Pop instruction involves wrong popping of values in thread stack
   1. Does armv7 cortex-a9 implements standard ARM procedure call
      1. Yes for now
   2. Need to know how compiler call function and return from it
      1. Not clear in docs , compiler flags which calling convention is used
      2. Note that , "Bl" by defaults save PC at LR which is (current+4)
      3. See simple/complex example with no parameters
         1. See table below
   3. Debug ,without stub works, the scenario of
      1. App task call function and this function cause it context switch
      2. And the restoring when it is resumed again
      3. Check to see how it return from this function call
      4. This scenario is anyfunction🡪OS\_Sched()🡪OS\_TASK\_SW() which is OSCtxSw() which just assembled as "bl 109f6c"

But return from it is just a ucos-ii's pop of all context

Note : this not match our case

* 1. Debug exactly what our case/scenario is
     1. OSTaskCreateExt(AppTaskStub, /\* Create the Task #3"stub". \*/

112de0: e3a03001 mov r3, #1

* + - 1. Monitor its calling steps
    1. Executes till the end of OSTaskCreateExt
       1. Monitor return steps
    2. During this ,we need to monitor the stack for current task which is apptask start via OSTCBHighRdy-> OSTCBStkPtr in memory view
  1. Note : ucos-ii stack growth is from high to low addresses
     1. Ucos-ii works as same direction as pop(LDMIA) and push(STMIA) ARM instructions
  2. How RT-thread handles pop instruction
     1. Does RT-thread stub environment differs from ours(exist of kernel context switch) and it is just exception handling routine

|  |  |  |  |
| --- | --- | --- | --- |
| Function call | Calling steps | Return steps |  |
| Math\_init(); | bl 10b04c <Math\_Init> | pop {fp, pc} |  |
| AppPrint(".") | movw r0, #21424 ; 0x53b0  movt r0, #17  bl 112ff4 <AppPrint> | sub sp, fp, #4  pop {fp, pc} |  |
| OSTimeDlyHMSM(0, 0, 0, 100); | mov r0, #0  mov r1, #0  mov r2, #0  mov r3, #100 ; 0x64  bl 108c3c <OSTimeDlyHMSM> | sub sp, fp, #4  pop {fp, pc} |  |
| OSTaskCreateExt(AppTaskStub,  (void \*)0,  (OS\_STK \*)&AppTask3Stk[APP\_CFG\_TASK\_1\_STK\_SIZE - 1],  (OS\_PRIO )APP\_CFG\_TASK\_3\_PRIO,  (OS\_PRIO )APP\_CFG\_TASK\_3\_PRIO,  (OS\_STK \*)&AppTask3Stk[0],  (CPU\_STK\_SIZE)APP\_CFG\_TASK\_3\_STK\_SIZE,  (void \*)0,  (CPU\_INT16U )(OS\_TASK\_OPT\_STK\_CHK | OS\_TASK\_OPT\_STK\_CLR)); | mov r3, #1  str r3, [sp]  movw r3, #59928 ; 0xea18  movt r3, #17  str r3, [sp, #4]  mov r3, #512 ; 0x200  str r3, [sp, #8]  mov r3, #0  str r3, [sp, #12]  mov r3, #3  str r3, [sp, #16]  movw r0, #12148 ; 0x2f74  movt r0, #17  mov r1, #0  ldr r2, [pc, #148] ; 112eb4 <AppTaskCreate+0xe0>  mov r3, #1  bl 107b90 <OSTaskCreateExt> | mov r0, r3  sub sp, fp, #4  pop {fp, pc} |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0x11d964 : 0x11D964 <Hex>

Address 0 - 3 4 - 7 8 - B C - F

0011D960 D09F1000 13010060 02000000 09000000

0011D970 C8F71100 04040404 05050505 06060606

0011D980 07070707 08080808 09090909 10101010

0011D990 ACD91100 12121212 601F1000 601F1000

601F1000

0011D9A0 13010020 13010060 E4D91100 9C7C1000

0011D9B0 00020000 00000000 03000000 10101010

0011D9C0 DCD91101 14F21100 00000000 70301100

0011D9D0 00000000 13010000 D8F11100 13010020

0011D9E0 04DA1100 1C2F1100 01000000 18EA1100

0011D9F0 00020000 00000000 03000000 13010020

0011DA00 14DA1100 A42E1100 11111111 00000000