Test / Comparison of CPU speed / Cache Performance AM3359 vs. AM6442 R5F

20260105 (fm)

Summary / remarks:

- CCS projects test_cpuspeed on both AM3359, AM 6442
- checked AM3359 Cortex A8 DDR3 vs. AM6442 R5F DDR4 CPU and cache performance
- both 800 MHz ARM v7 architecture
- main SoC difference: 256 KByte L2 cache is available for AM3359 Cortex A8, but not for AM6442 R5F
- no exact comparison because of different compilers
- see remarks on measurements

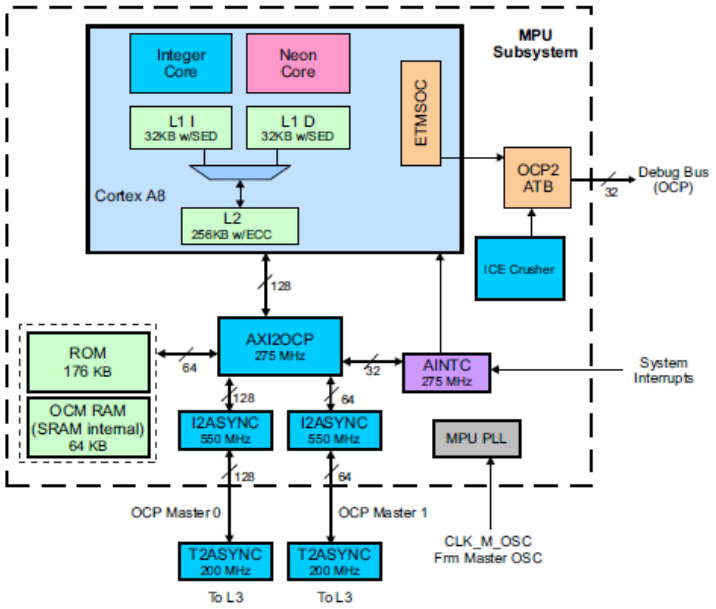
Architecture comparison

AM 3359: (SPRS717J –OCTOBER 2011–REVISED APRIL 2016)

1.1 Features

• Up to 1-GHz Sitara™ ARM® Cortex®-A8 32-Bit RISC Processor
– NEON™ SIMD Coprocessor
– 32KB of L1 Instruction and 32KB of Data Cache With Single-Error Detection (Parity)
– 256KB of L2 Cache With Error Correcting Code (ECC)
– 176KB of On-Chip Boot ROM
– 64KB of Dedicated RAM
– Emulation and Debug – JTAG
• On-Chip Memory (Shared L3 RAM)
– 64KB of General-Purpose On-Chip Memory Controller (OCMC) RAM
– Accessible to All Masters
• External Memory Interfaces (EMIF)
– mDDR(LPDDR), DDR2, DDR3, DDR3L Controller:
• mDDR: 200-MHz Clock (400-MHz Data Rate)
• DDR2: 266-MHz Clock (532-MHz Data Rate)
• DDR3: 400-MHz Clock (800-MHz Data Rate)
• DDR3L: 400-MHz Clock (800-MHz Data Rate)
• 16-Bit Data Bus
• 1GB of Total Addressable Space

Figure 3-1. Microprocessor Unit (MPU) Subsystem



AM6442: (SPRSP56F – JANUARY 2021 – REVISED OCTOBER 2023)

1 Features

Processor cores:

• 1× Dual 64-bit Arm® Cortex®-A53 microprocessor subsystem at up to 1.0 GHz
– Dual-core Cortex-A53 cluster with 256KB L2 shared cache with SECDED ECC
– Each A53 Core has 32KB L1 DCache with SECDED ECC and 32KB L1 ICache with Parity protection
• Up to 2× Dual-core Arm® Cortex®-R5F MCU subsystems at up to 800 MHz, integrated for realtime processing
– Dual-core Arm® Cortex®-R5F supports dualcore and single-core modes
– 32KB ICache, 32KB DCache and 64KB TCM per each R5F core for a total of 256KB TCM with SECDED ECC on all memories
• 1× Single-core Arm® Cortex®-M4F MCU at up to 400 MHz
– 256KB SRAM with SECDED ECC

6.1.3.1 A53SS Block Diagram

Figure 6-2 shows the block diagram of the A53SS.

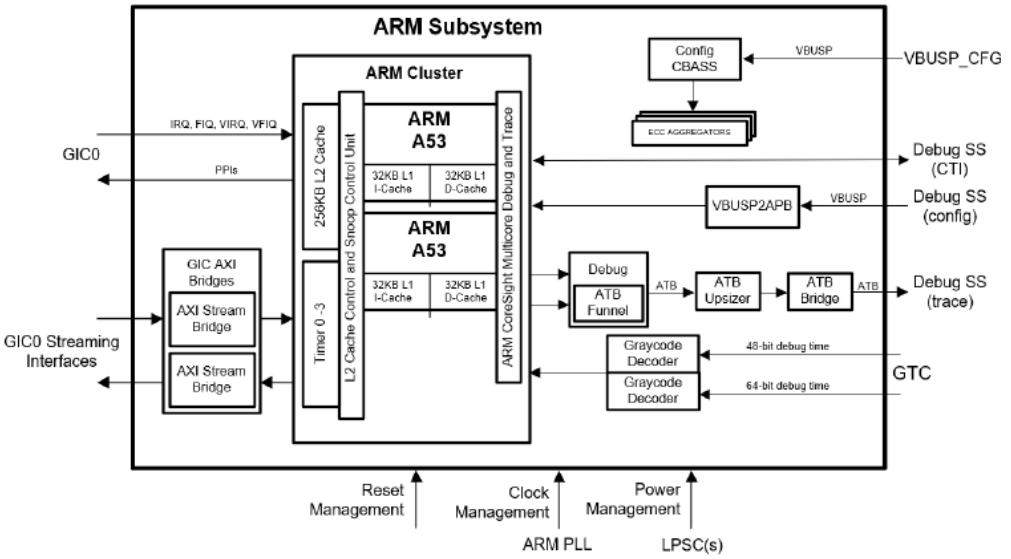


Figure 6-2. A53SS Block Diagram

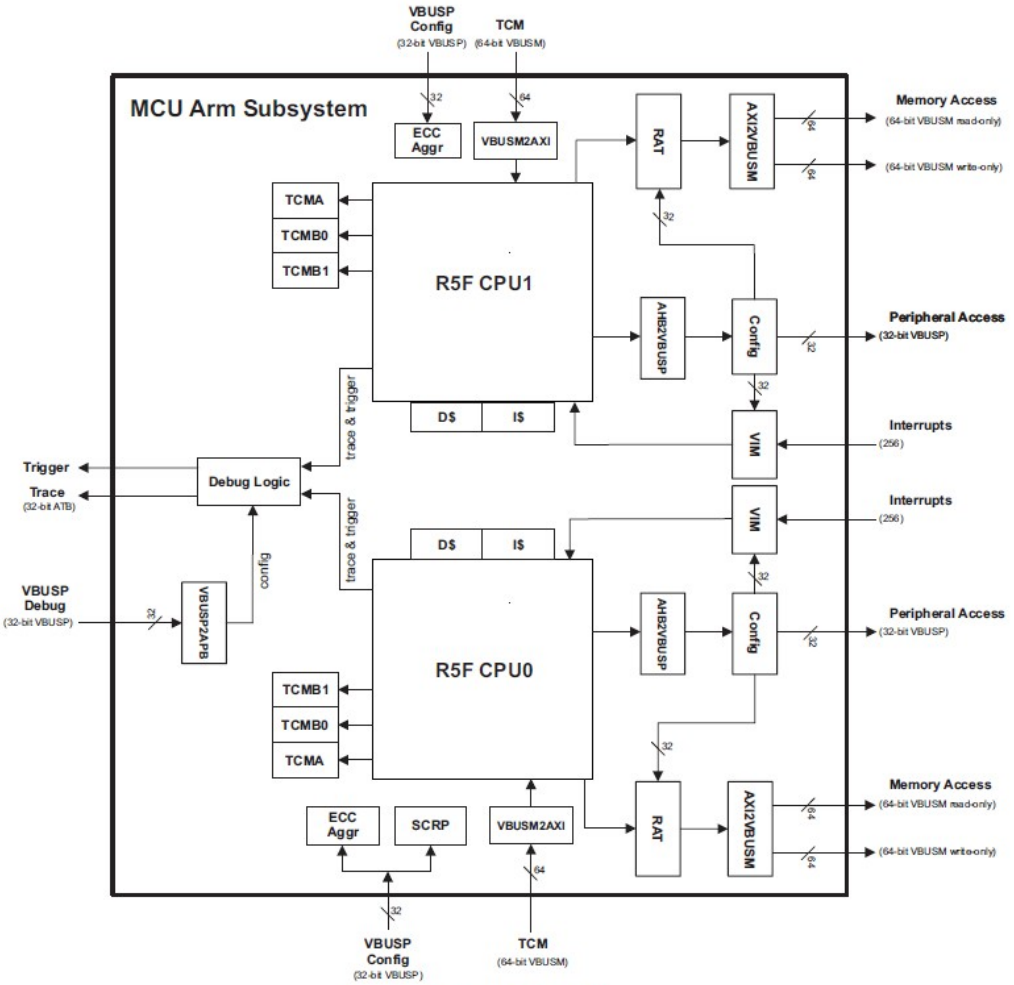
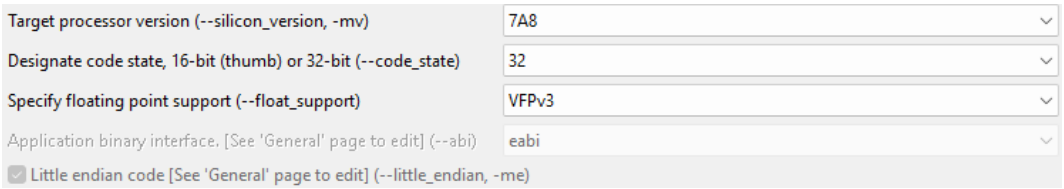
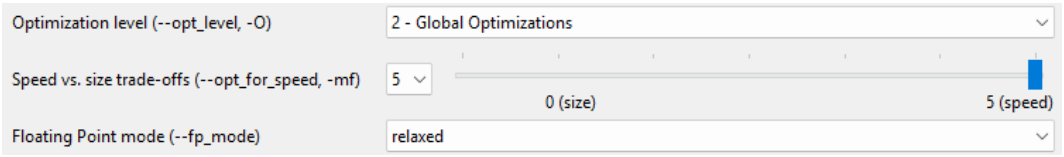


Figure 6-53. R5FSS Block Diagram

AM3359






```
MEMORY CONFIGURATION

         name            origin    length      used     unused   attr    fill
----------------------  --------  ---------  --------  --------  ----  --------
  SRAM_LO               402f0000   00000400  00000000  00000400  RW X
  SRAM_HI               402f0400   0000fc00  00000000  0000fc00  RW X
  OCMC_SRAM             40300000   00010000  00000000  00010000  RW X
  DDR3                  80000000   04000000  0003c39d  03fc3c63  RWIX
```

Measured CPU timer values in nanoseconds:

```
[0]     /* test #1: write memory with 8192 uint32_t */
[1]     /* test #2: float operations, write some memory with 2048 real64_t */
[2]     /* first call nops256() */
[3]     /* 127 * (256 nops + call / return / loop) = 32768 nops, probably cached */
[4]     /* call nops32768(), 32728 nops, too big for L1cache */
[5]     /* sum of all times */
```

tsdiff	double[10]	[26433.75,969...
(x)= [0]	double	26433.75
(x)= [1]	double	96902.5
(x)= [2]	double	4811.25
(x)= [3]	double	41150.0
(x)= [4]	double	592203.75
(x)= [5]	double	761501.25

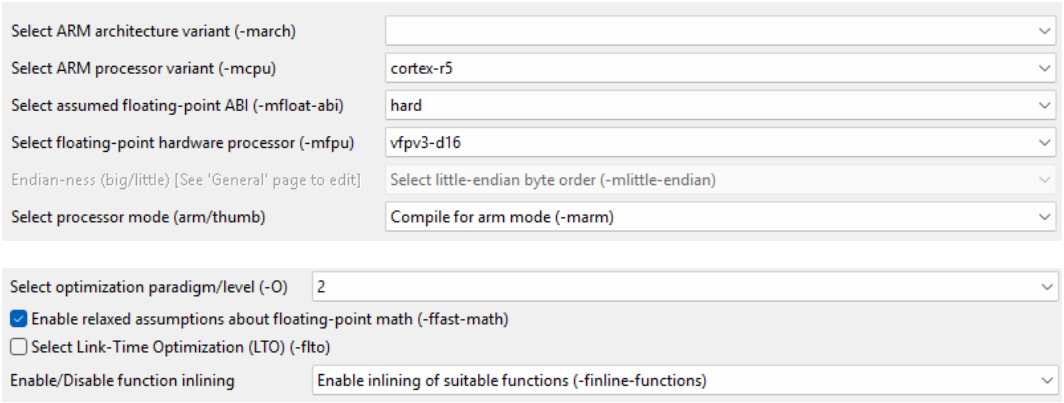
1st run

tsdiff	double[10]	[25760.0,9641...
(x)= [0]	double	25760.0
(x)= [1]	double	96413.75
(x)= [2]	double	447.5
(x)= [3]	double	40882.5
(x)= [4]	double	51747.5
(x)= [5]	double	215251.25

2nd run

- small diffs in [0] and [1]
- nop256() and nop32768() already (at least) in L2 cache
>> 41000 / 127 = 322.83 ns, 256 * 1.25 ns = 320 ns, i. e. 1 cycle per nop
- repeated nop32768 in less than 1/10 of 1st call
- >> 51750 / 1.25 = 41400 cycles for 32768 nops

AM6442



.bss, .text in DDR4:

```
MEMORY CONFIGURATION

         name             origin    length      used     unused   attr    fill
----------------------  --------  ---------  --------  --------  ----  --------
  R5F_VECS              00000000   00000040  00000040  00000000  RWIX
  R5F_TCMA              00000040   00007fc0  00000000  00007fc0  RWIX
  R5F_TCMB0             41010000   00008000  00000000  00008000  RWIX
  FLASH                 60100000   00080000  00000000  00080000  RWIX
  NON_CACHE_MEM         70060000   00008000  00000000  00008000  RWIX
  MSRAM                 70080000   00040000  00004e28  0003b1d8  RWIX
  USER_SHM_MEM          701d0000   00000080  00000000  00000080  RWIX
  LOG_SHM_MEM           701d0080   00003f80  00000000  00003f80  RWIX
  RTOS_NORTOS_IPC_SHM_M 701d4000   0000c000  00000000  0000c000  RWIX
  DDR_ALL               80000000   40000000  00059c80  3ffa6380  RWIX


SEGMENT ALLOCATION MAP

run origin  load origin   length   init length attrs members
----------  ----------- ---------- ----------- ----- -------
00000000    00000000    00000040   00000040    r-x
  00000000    00000000    00000040   00000040    r-x .vectors
70080000    70080000    000024c0   000024c0    rw-
  70080000    70080000    000024c0   000024c0    rw- .data
700824c0    700824c0    00002968   00001568    r-x
  700824c0    700824c0    00000c20   00000c20    r-x .text.hwi
  700830e0    700830e0    000004a0   000004a0    r-x .text.cache
  70083580    70083580    000002d0   000002d0    r-x .text.mpu
  70083850    70083850    000001d0   000001d0    r-x .text.boot
  70083a20    70083a20    00000008   00000008    r-x .text:abort
  70083a28    70083a28    00000100   00000000    r-- .irqstack
  70083b28    70083b28    00000100   00000000    r-- .fiqstack
  70083c28    70083c28    00001000   00000000    r-- .svcstack
  70084c28    70084c28    00000100   00000000    r-- .abortstack
  70084d28    70084d28    00000100   00000000    r-- .undefinedstack
80000000    80000000    0003a980   0003a980    r-x
  80000000    80000000    00038d00   00038d00    r-x .text
  80038d00    80038d00    00001c80   00001c80    r-- .rodata
8003a980    8003a980    0001f300   00000000    rw-
  8003a980    8003a980    00013300   00000000    rw- .bss
  8004dc80    8004dc80    00008000   00000000    rw- .sysmem
  80055c80    80055c80    00004000   00000000    rw- .stack
```

tsdiff	double[10]	[67240.0,887...
[0]	double	67240.0
[1]	double	88728.0
[2]	double	7456.0
[3]	double	41528.0
[4]	double	1004880.0
[5]	double	1209832.0

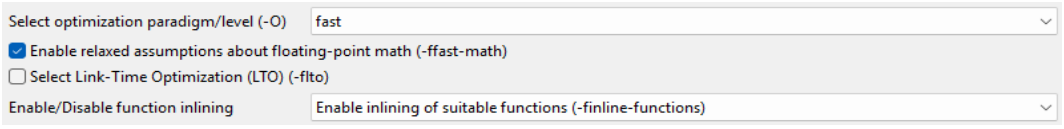
1st run – nop32768: 1 210 000 ns vs. 592 000 ns – about twice the time

tsdiff	double[10]	[68136.0,936...
[0]	double	68136.0
[1]	double	93696.0
[2]	double	4960.0
[3]	double	41296.0
[4]	double	954920.0
[5]	double	1163008.0

2nd run – 1 163 000 vs. 51747 – about factor 20! (no L2 cache)

[0]: 68000 ns vs. 25000 ns – other optimisation (see below)
[1]: (float ops) nearly similar

-Ofast:



tsdiff	double[10]	[30200.0,107...
[0]	double	30200.0
[1]	double	107296.0
[2]	double	7552.0
[3]	double	41744.0
[4]	double	1004816.0

1st run

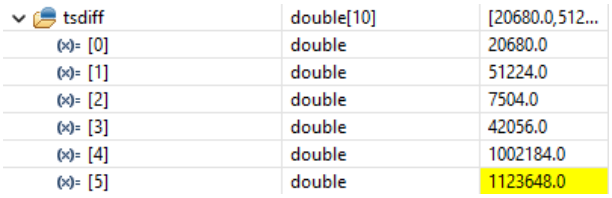
tsdiff	double[10]	[30200.0,107...
[0]	double	30200.0
[1]	double	107296.0
[2]	double	7552.0
[3]	double	41744.0
[4]	double	1004816.0
[5]	double	1191608.0

2nd run

- [0] better, but not reaching AM3359 performance

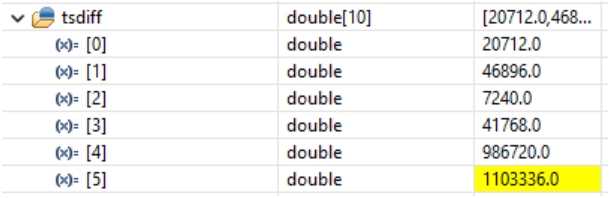
-Ofast, data in MSRAM

```
MEMORY CONFIGURATION

         name             origin    length      used     unused   attr    fill
----------------------  --------  ---------  --------  --------  ----  --------
  R5F_VECS              00000000  00000040  00000040  00000000  RWIX
  R5F_TCMA              00000040  00007fc0  00000000  00007fc0  RWIX
  R5F_TCMB0             41010000  00008000  00000000  00008000  RWIX
  FLASH                 60100000  00080000  00000000  00080000  RWIX
  NON_CACHE_MEM         70060000  00008000  00000000  00008000  RWIX
  MSRAM                 70080000  00040000  00024128  0001bed8  RWIX
  USER_SHM_MEM          701d0000  00000080  00000000  00000080  RWIX
  LOG_SHM_MEM           701d0080  00003f80  00000000  00003f80  RWIX
  RTOS_NORTOS_IPC_SHM_M 701d4000  0000c000  00000000  0000c000  RWIX
  DDR_ALL               80000000  40000000  0003a9c0  3ffc5640  RWIX

SEGMENT ALLOCATION MAP

run origin  load origin   length   init length attrs members
----------  -----------  ---------- ----------- ----- -------
00000000    00000000    00000040   00000040    r-x
  00000000    00000000    00000040   00000040    r-x .vectors
70080000    70080000    0001f300   00000000    rw-
  70080000    70080000    00013300   00000000    rw- .bss
  70093300    70093300    00008000   00000000    rw- .sysmem
  7009b300    7009b300    00004000   00000000    rw- .stack
7009f300    7009f300    000024c0   000024c0    rw-
  7009f300    7009f300    000024c0   000024c0    rw- .data
700a17c0    700a17c0    00002968   00001568    r-x
  700a17c0    700a17c0    00000c20   00000c20    r-x .text.hwi
  700a23e0    700a23e0    000004a0   000004a0    r-x .text.cache
  700a2880    700a2880    000002d0   000002d0    r-x .text.mpu
  700a2b50    700a2b50    000001d0   000001d0    r-x .text.boot
  700a2d20    700a2d20    00000008   00000008    r-x .text:abort
  700a2d28    700a2d28    00000100   00000000    r-- .irqstack
  700a2e28    700a2e28    00000100   00000000    r-- .fiqstack
  700a2f28    700a2f28    00001000   00000000    r-- .svcstack
  700a3f28    700a3f28    00000100   00000000    r-- .abortstack
  700a4028    700a4028    00000100   00000000    r-- .undefinedstack
80000000    80000000    0003a9c0   0003a9c0    r-x
  80000000    80000000    00038d40   00038d40    r-x .text
  80038d40    80038d40    00001c80   00001c80    r-- .rodata
```



tsdiff	double[10]	[20680.0,512...
(x)= [0]	double	20680.0
(x)= [1]	double	51224.0
(x)= [2]	double	7504.0
(x)= [3]	double	42056.0
(x)= [4]	double	1002184.0
(x)= [5]	double	1123648.0

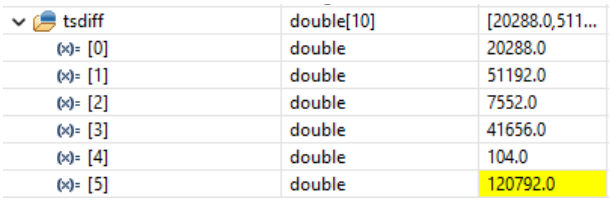
1st run



tsdiff	double[10]	[20712.0,468...
(x)= [0]	double	20712.0
(x)= [1]	double	46896.0
(x)= [2]	double	7240.0
(x)= [3]	double	41768.0
(x)= [4]	double	986720.0
(x)= [5]	double	1103336.0

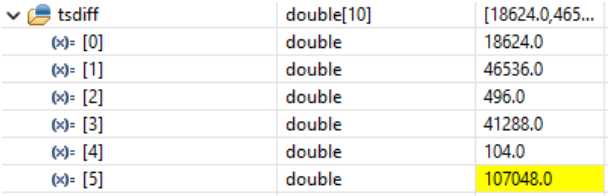
2nd run
- [0] better, beating AM3359 performance
- [1] (float) about half the AM3359 time, not clear why

no nop32768() call



tsdiff	double[10]	[20288.0,511...
[0]	double	20288.0
[1]	double	51192.0
[2]	double	7552.0
[3]	double	41656.0
[4]	double	104.0
[5]	double	120792.0

1st run



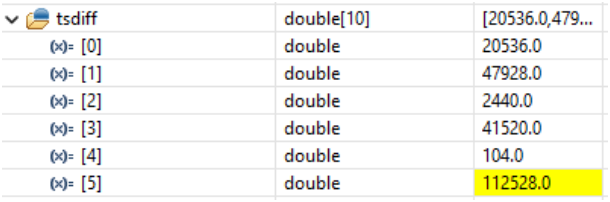
tsdiff	double[10]	[18624.0,465...
[0]	double	18624.0
[1]	double	46536.0
[2]	double	496.0
[3]	double	41288.0
[4]	double	104.0
[5]	double	107048.0

2nd run

- nop256() remains in L1 cache, 1st call faster

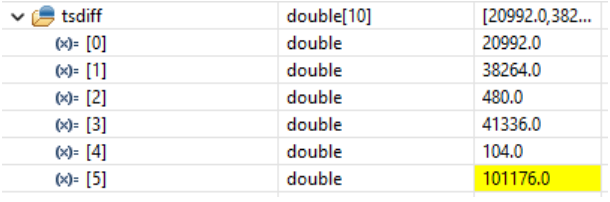
code and data in MSRAM

```
MEMORY CONFIGURATION

         name             origin    length      used     unused   attr    fill
----------------------  --------  ---------  --------  --------  ----  --------
  R5F_VECS              00000000   00000040  00000040  00000000  RWIX
  R5F_TCMA              00000040   00007fc0  00000000  00007fc0  RWIX
  R5F_TCMB0             41010000   00008000  00000000  00008000  RWIX
  FLASH                 60100000   00080000  00000000  00080000  RWIX
  NON_CACHE_MEM         70060000   00008000  00000000  00008000  RWIX
  MSRAM                 70080000   00040000  00034ab8  0000b548  RWIX
  USER_SHM_MEM          701d0000   00000080  00000000  00000080  RWIX
  LOG_SHM_MEM           701d0080   00003f80  00000000  00003f80  RWIX
  RTOS_NORTOS_IPC_SHM_M 701d4000   0000c000  00000000  0000c000  RWIX
  DDR_ALL               80000000   40000000  00000000  40000000  RWIX

SEGMENT ALLOCATION MAP

run origin  load origin   length   init length attrs members
----------  -----------  ---------- ----------- ----- -------
00000000    00000000    00000040   00000040    r-x
  00000000    00000000    00000040   00000040    r-x .vectors
70080000    70080000    0001a9a0   0001a9a0    r-x
  70080000    70080000    00018d20   00018d20    r-x .text
  70098d20    70098d20    00001c80   00001c80    r-- .rodata
7009a9a0    7009a9a0    00015300   00000000    rw-
  7009a9a0    7009a9a0    00013300   00000000    rw- .bss
  700adca0    700adca0    00001000   00000000    rw- .sysmem
  700aeca0    700aeca0    00001000   00000000    rw- .stack
700afca0    700afca0    000024c0   000024c0    rw-
  700afca0    700afca0    000024c0   000024c0    rw- .data
700b2160    700b2160    00002958   00001558    r-x
  700b2160    700b2160    00000c20   00000c20    r-x .text.hwi
  700b2d80    700b2d80    000004a0   000004a0    r-x .text.cache
  700b3220    700b3220    000002d0   000002d0    r-x .text.mpu
  700b34f0    700b34f0    000001c0   000001c0    r-x .text.boot
  700b36b0    700b36b0    00000008   00000008    r-x .text:abort
  700b36b8    700b36b8    00000100   00000000    r-- .irqstack
  700b37b8    700b37b8    00000100   00000000    r-- .fiqstack
  700b38b8    700b38b8    00001000   00000000    r-- .svcstack
  700b48b8    700b48b8    00000100   00000000    r-- .abortstack
  700b49b8    700b49b8    00000100   00000000    r-- .undefinedstack
```



tsdiff	double[10]	[20536.0,479...
(x)= [0]	double	20536.0
(x)= [1]	double	47928.0
(x)= [2]	double	2440.0
(x)= [3]	double	41520.0
(x)= [4]	double	104.0
(x)= [5]	double	112528.0

1st run
- faster cache 1st of nop256() compared to DDR4



tsdiff	double[10]	[20992.0,382...
(x)= [0]	double	20992.0
(x)= [1]	double	38264.0
(x)= [2]	double	480.0
(x)= [3]	double	41336.0
(x)= [4]	double	104.0
(x)= [5]	double	101176.0

2nd run