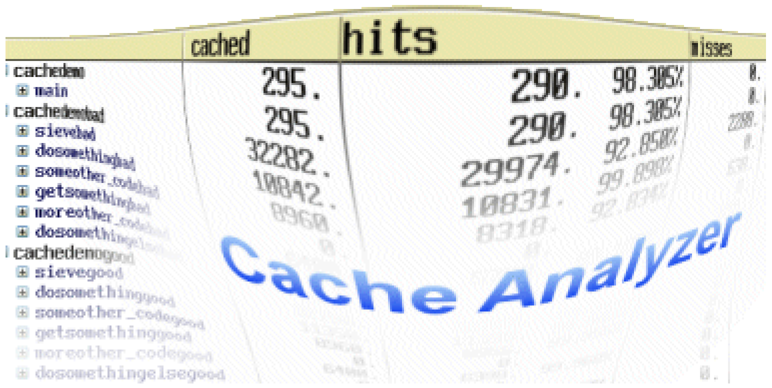
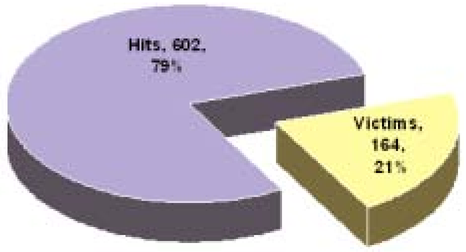
Trace32 _Cache性能分析

- Cache 介绍
- Trace32工具下Cache性能分析





Trace32 _Cache性能分析

- ➢ Cache 介绍
- ■ Trace32 下Cache性能分析

Cache 介绍

- Cache工作原理
- Cache性能分析

Cache 工作原理

- Cache 又叫高速缓冲寄存器，位于CPU与内存之间，是一种特殊的存储器子系统。

- 程序执行的局部性：

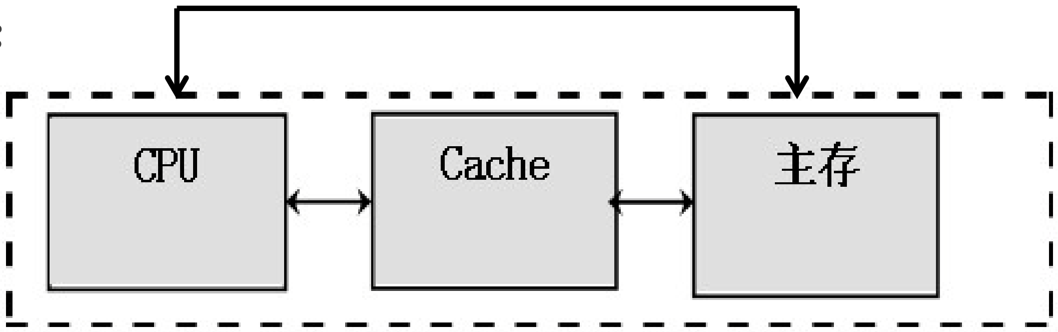
  1. 时间局部性：最近访问过的代码可能是不久将被访问的代码。

  2. 空间局部性：地址相邻的代码可能会被一起访问

  即：对局部存储器地址频繁访问，而对此地址范围外访问甚少的现象

- 根据局部性原理，可以在主存和CPU之间设置一个高速的，容量相对较少的存储器，如果当前正在执行的程序和数据存放在这个存取器中，则当程序运行时，不必从主存存取指令和取数据，而是直接访问这个高速存储器即可，提高程序的执行速度，这个存储器称为高速缓冲存取器即Cache.

- Cache-主存层次：



- CPU在访问内存时，首先判断所要访问的内容是否在cache中，如果在，则称为命中（hit），此时CPU直接从cache中调用该内容；否则称为未命中（miss），CPU会通过cache对主存中的相应内容进行操作。

- 命中率：CPU要访问的信息在Cache 中的比率
- 未命中率：CPU要访问的信息不在Cache 中的比率

Cache 性能分析

- Cache 系统加速比Sp

运用Cache 技术主要目的是提高CPU对存取器的访问速度，加速比是其重要的性能参数。

$$S_p = \frac{T_m}{T} = \frac{T_m}{H \cdot T_c + (1-H) \cdot T_m} = \frac{1}{(1-H) + H \cdot \frac{T_c}{T_m}}) = f(H, \frac{T_m}{T_c})$$

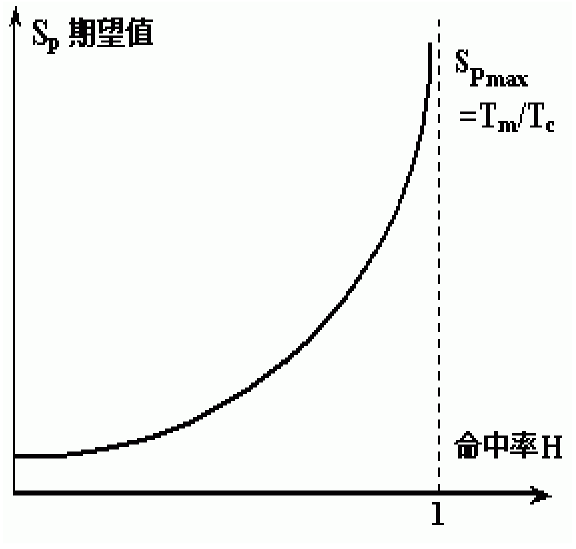
其中： $T = H \times T_c + (1-H) \times T_m$

T_m为主存储器的访问周期，T_c为Cache的访问周期，T则为Cache存储系统的等效访问周期，H为命中率。

- 可以看出，加速比的大小与两个因素有关：命中率H及Cache与主存访问周期的比值T_c/T_m。一旦器件选定,主存访问周期的比值T_c/T_m 基本确定。

  所以通常采用 “命中率”来测量cache的效率，命中率越高加速比越大。

- 右图显示Sp与H关系：



举例说明

- 例如：假设RAM的存取时间为8ns，CACHE的存取时间为1ns:

若命中率H =95%: T = 1ns×95% + 8ns×5% = 1.35ns

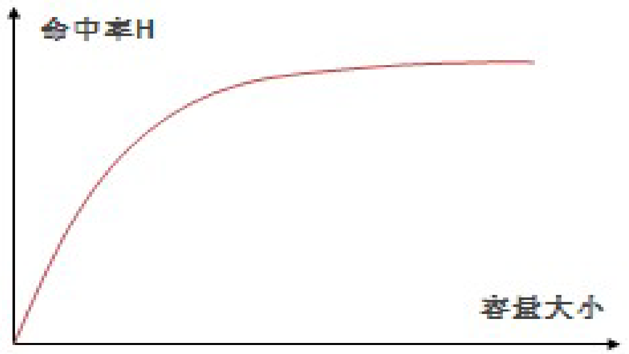
若命中率H =90%：T = 1ns×90% + 8ns×10% = 1.7ns

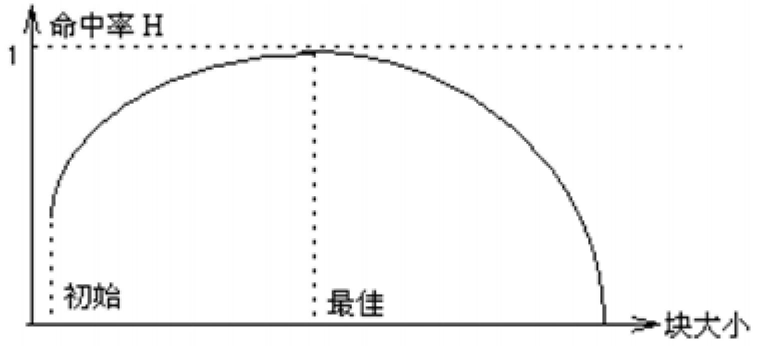
若命中率H =60%：T = 1ns×60% + 8ns×40% = 3.8ns

T:系统的平均存取速度≈Cache存取速度×命中率 +RAM存取速度×不命中率

影响Cache 命中率的因素

- Cache的容量
- 块的大小
- 映象方式
- 替换算法
- **程序执行中地址流的分布**

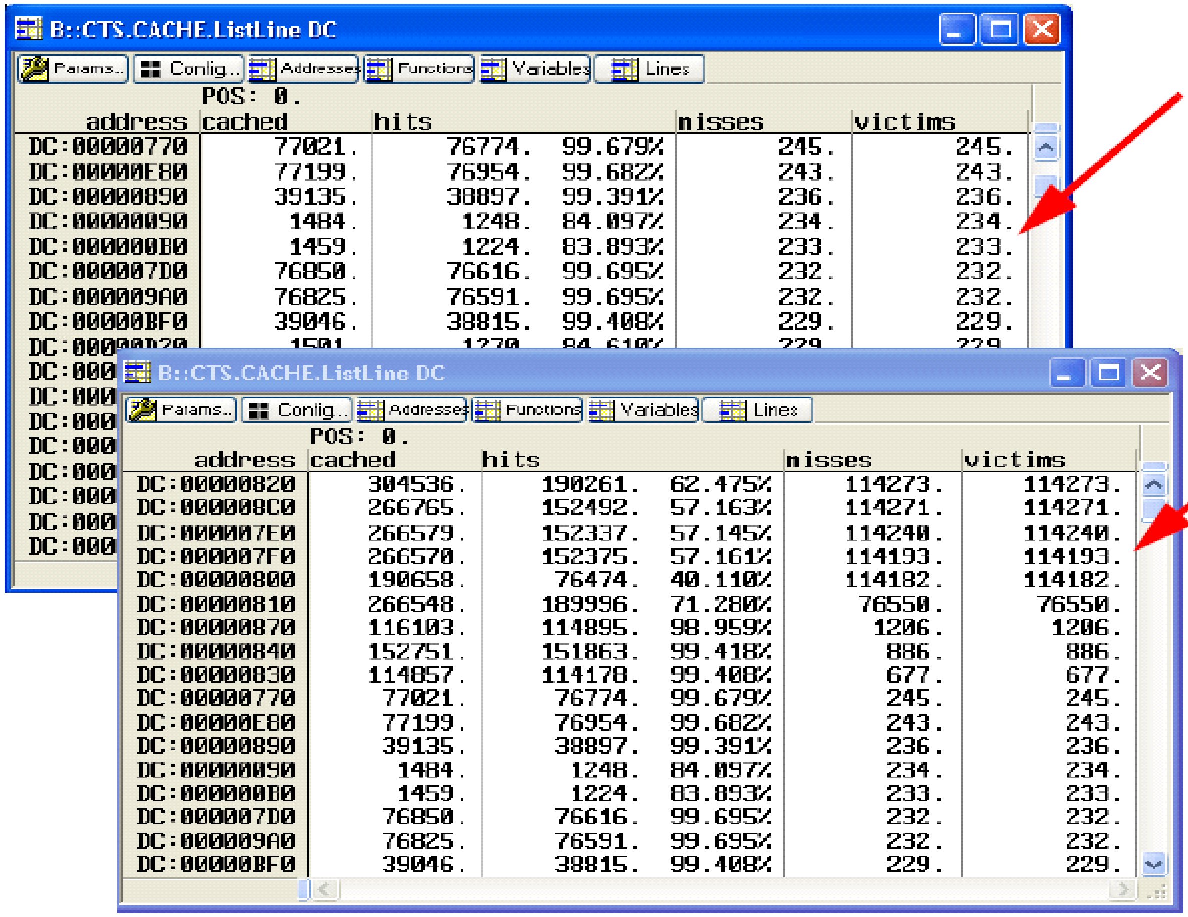




Trace32 下Cache性能分析

- Cache 介绍
- Trace32下Cache性能分析

缓存使用率分析：



B::CTS.CACHE.ListLine DC

```
            POS: 0.
    address cached      hits                    misses      victims
DC:00000770     77021.     76774.   99.679%        245.        245.
DC:00000E80     77199.     76954.   99.682%        243.        243.
DC:00000890     39135.     38897.   99.391%        236.        236.
DC:00000090      1484.      1248.   84.097%        234.        234.
DC:000000B0      1459.      1224.   83.893%        233.        233.
DC:000007D0     76850.     76616.   99.695%        232.        232.
DC:000009A0     76825.     76591.   99.695%        232.        232.
DC:00000BF0     39046.     38815.   99.408%        229.        229.
DC:000
DC:000
DC:000
DC:000
DC:000
DC:000
DC:000
DC:000
```

If the cache is used effectively the number of cache victims is evenly distributed

B::CTS.CACHE.ListLine DC

```
            POS: 0.
    address cached      hits                    misses      victims
DC:00000820    304536.    190261.   62.475%     114273.     114273.
DC:000008C0    266765.    152492.   57.163%     114271.     114271.
DC:000007E0    266579.    152337.   57.145%     114240.     114240.
DC:000007F0    266570.    152375.   57.161%     114193.     114193.
DC:00000800    190658.     76474.   40.110%     114182.     114182.
DC:00000810    266548.    189996.   71.280%      76550.      76550.
DC:00000870    116103.    114895.   98.959%       1206.       1206.
DC:00000840    152751.    151863.   99.418%        886.        886.
DC:00000830    114857.    114178.   99.408%        677.        677.
DC:00000770     77021.     76774.   99.679%        245.        245.
DC:00000E80     77199.     76954.   99.682%        243.        243.
DC:00000890     39135.     38897.   99.391%        236.        236.
DC:00000090      1484.      1248.   84.097%        234.        234.
DC:000000B0      1459.      1224.   83.893%        233.        233.
DC:000007D0     76850.     76616.   99.695%        232.        232.
DC:000009A0     76825.     76591.   99.695%        232.        232.
DC:00000BF0     39046.     38815.   99.408%        229.        229.
```

If the cache is used ineffectively the number of cache victims is unbalanced

Trace32工具下Cache性能分析

- Trace32 tool: PowerTrace
- Cache性能分析目的：验证cache的使用效率
- Cache性能分析前提：

  instruction cache Analysis:

    Program flow is sampled into the trace buffer

  data cache Analysis:

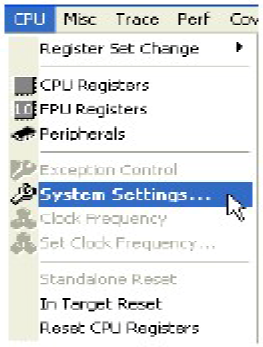
    At least the address for the read and write accesses are sampled into the trace buffer

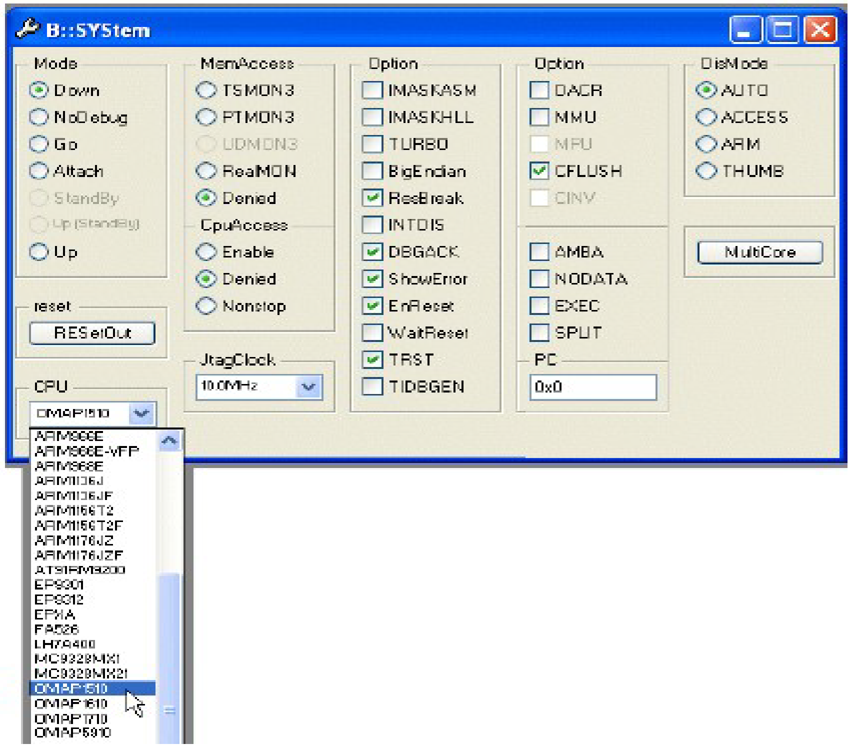
- Cache analysis is a part of CTS functionality
- Recommand：

  use the trace in FIFO or LEASH mode
- no trace filters should be used

定义Cache Structure

- 1. 在SYSTEM window 选择CPU




B::SYStem

Mode: [✓ Down] NoDebug Go Attach StandBy Up (StandBy) Up

MemAccess: TSMON3 PTMON3 UDMON3 RealMON [✓ Denied]

CpuAccess: Enable [✓ Denied] Nonstop

reset: RESetOut

JtagClock: 10.0MHz

CPU: OMAP1510

Option: IMASKASM ☐, IMASKHLL ☐, TURBO ☐, BigEndian ☐, ResBreak ✓, INTDIS ☐, DBGACK ✓, ShowError ✓, EnReset ✓, WaitReset ☐, TRST ✓, TIDBGEN ☐

Option: DACR ☐, MMU ☐, MPU ☐, CFLUSH ✓, CINV ☐, AMBA ☐, NODATA ☐, EXEC ☐, SPLIT ☐

PC: 0x0

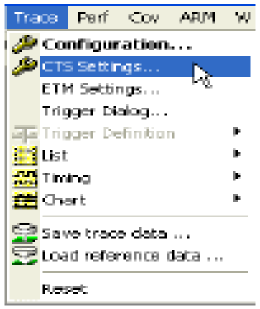
DisMode: [✓ AUTO] ACCESS ARM THUMB

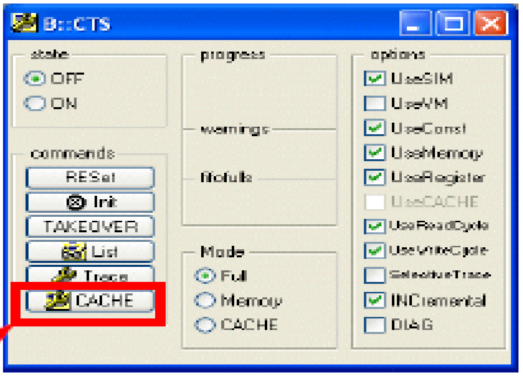
MultiCore

ARM966E
ARM966E-VFP
ARM968E
ARM1136J
ARM1136JF
ARM1156T2
ARM1156T2F
ARM1176JZ
ARM1176JZF
AT91RM9200
EP9301
EP9312
EPXA
FA526
LH7A400
MC9328MX1
MC9328MX21
OMAP1510
OMAP1610
OMAP1710
OMAP5910

CPU selection

2.打开CACHE.VIEW window



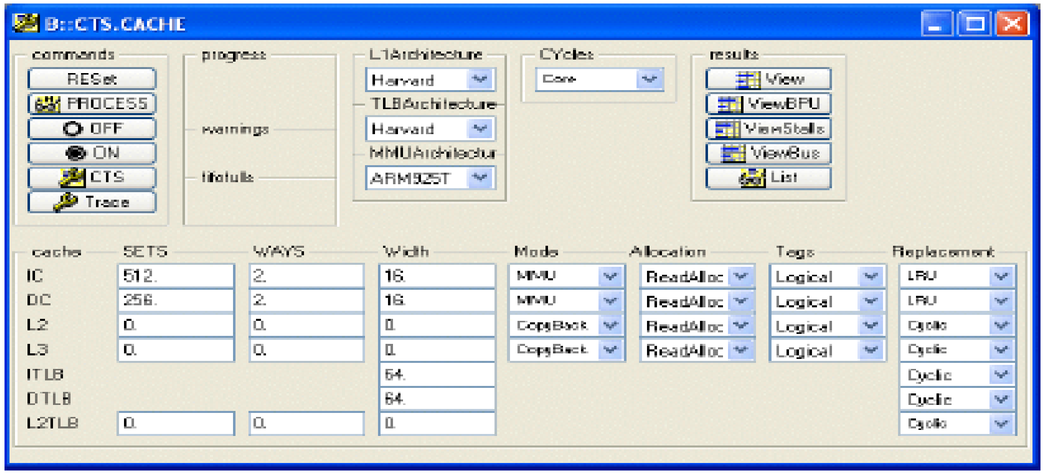


state: [✓ OFF] ON

Mode: [✓ Full] Memory CACHE

options:
- [✓] UseSIM
- [ ] UseVM
- [✓] UseConst
- [✓] UseMemory
- [✓] UseRegister
- [ ] UseCACHE
- [✓] UseReadCycle
- [✓] UseWriteCycle
- [ ] SelectiveTrace
- [✓] INCremental
- [ ] DIAG

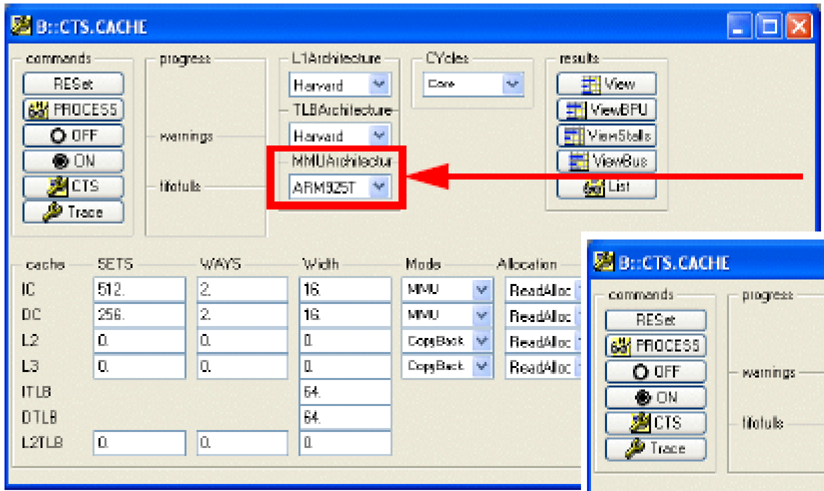
Display the definition of the cache structure



L1Architecture: Harvard
TLBArchitecture: Harvard
MMUArchitectur: ARM925T
CYcles: Core

cache	SETS	WAYS	Width	Mode	Allocation	Tags	Replacement
IC	512.	2.	16.	MMU	ReadAlloc	Logical	LRU
DC	256.	2.	16.	MMU	ReadAlloc	Logical	LRU
L2	0.	0.	0.	CopyBack	ReadAlloc	Logical	Cyclic
L3	0.	0.	0.	CopyBack	ReadAlloc	Logical	Cyclic
ITLB			64.				Cyclic
DTLB			64.				Cyclic
L2TLB	0.	0.	0.				Cyclic

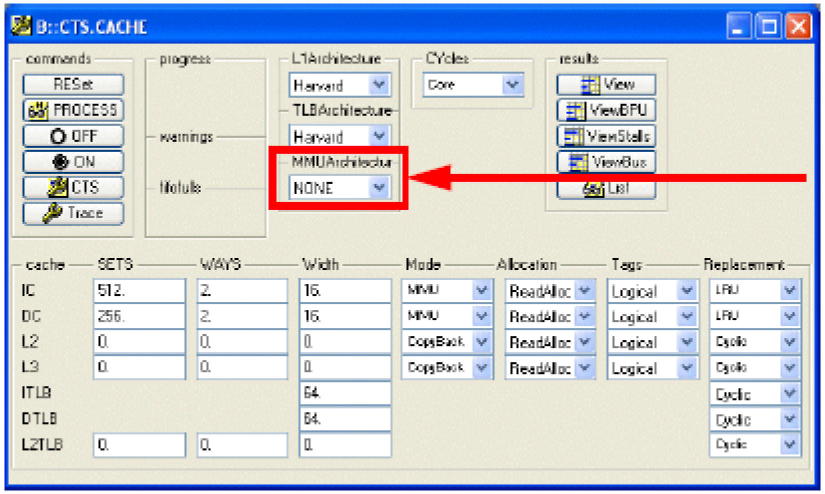
3.Define MMU Architecture



L1Architecture: Harvard
TLBArchitecture: Harvard
MMUArchitectur: ARM925T
CYcles: Core

cache	SETS	WAYS	Width	Mode	Allocation
IC	512.	2.	16.	MMU	ReadAlloc
DC	256.	2.	16.	MMU	ReadAlloc
L2	0.	0.	0.	CopyBack	ReadAlloc
L3	0.	0.	0.	CopyBack	ReadAlloc
ITLB			64.		
DTLB			64.		
L2TLB	0.	0.	0.		

Define the MMUArchitecture



L1Architecture: Harvard
TLBArchitecture: Harvard
MMUArchitectur: NONE
CYcles: Core

cache	SETS	WAYS	Width	Mode	Allocation	Tags	Replacement
IC	512.	2.	16.	MMU	ReadAlloc	Logical	LRU
DC	256.	2.	16.	MMU	ReadAlloc	Logical	LRU
L2	0.	0.	0.	CopyBack	ReadAlloc	Logical	Cyclic
L3	0.	0.	0.	CopyBack	ReadAlloc	Logical	Cyclic
ITLB			64.				Cyclic
DTLB			64.				Cyclic
L2TLB	0.	0.	0.				Cyclic

MMUArchitecture is set to NONE

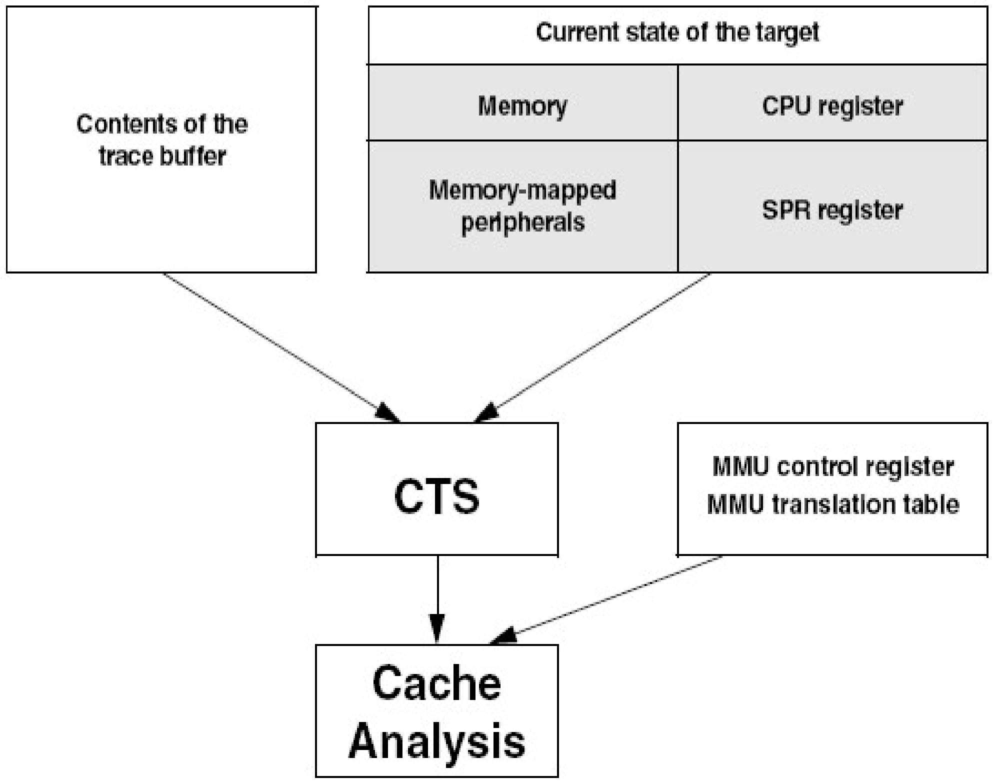
Note:set to NONE if:

1.The cache analysis is not full implemented for the select CPU

2.For the select CPU ,a pro/data flow can only be sampled to the trace after the cache was off

单个函数Cache 性能分析

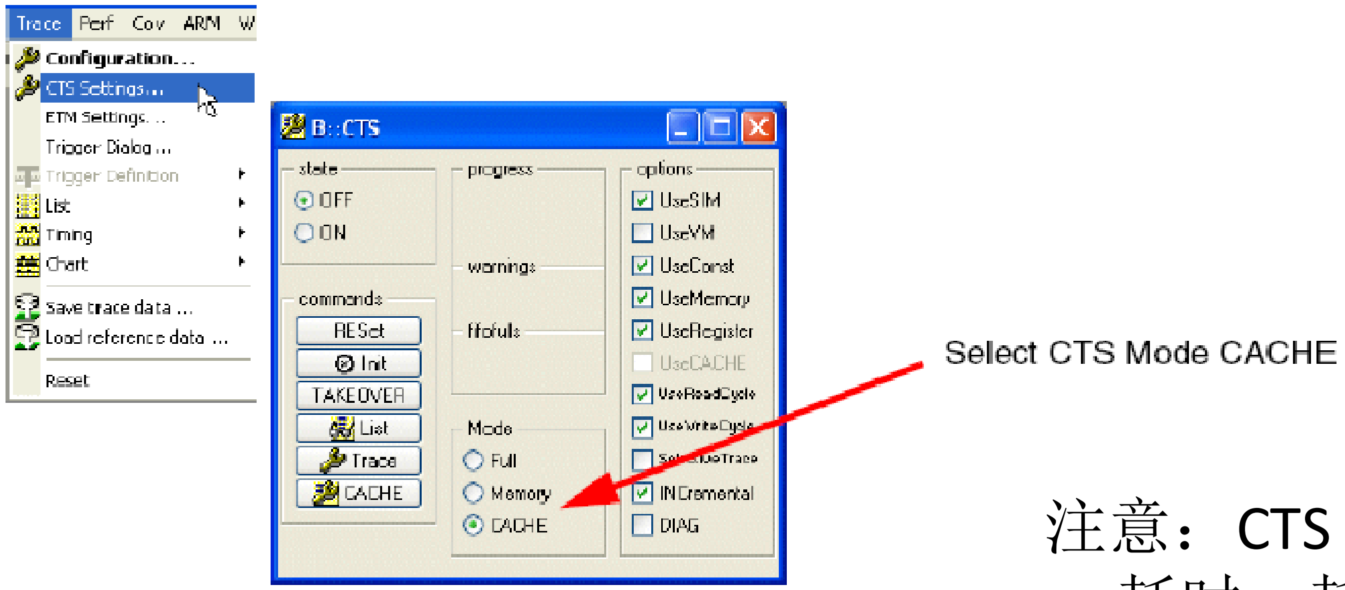
- 示意图：



- The prog/data flow sampled to the trace
- The current state of the target

Genenral Analysis

- 1. Select the MODE CACHE in the CTS window



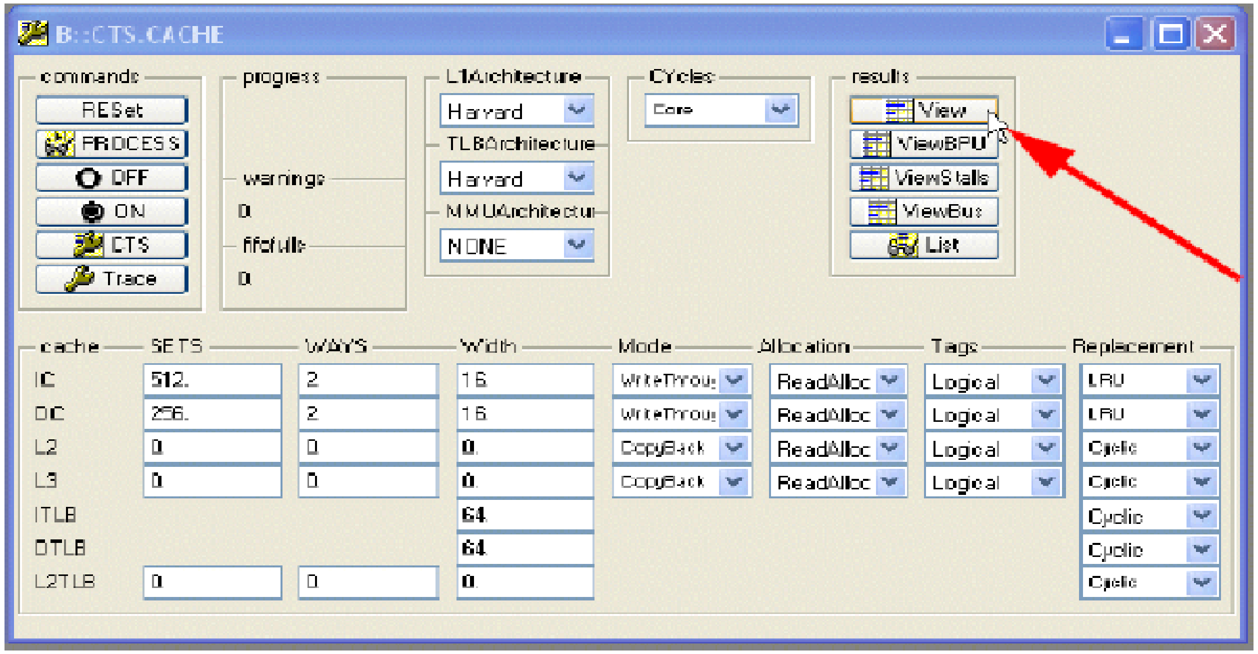
state: [✓ OFF] ON

Mode: Full Memory [✓ CACHE]

options: [✓ UseSIM] UseVM [✓ UseConst] [✓ UseMemory] [✓ UseRegister] UseCACHE [✓ UseReadCycle] [✓ UseWriteCycle] SelectiveTrace [✓ INCremental] DIAG

注意：CTS Mode Cache
耗时、耗内存

- 2. Push the VIEW button in the CTS.CACHE window



progress

warnings: 0

fifofulls: 0

L1Architecture: Harvard

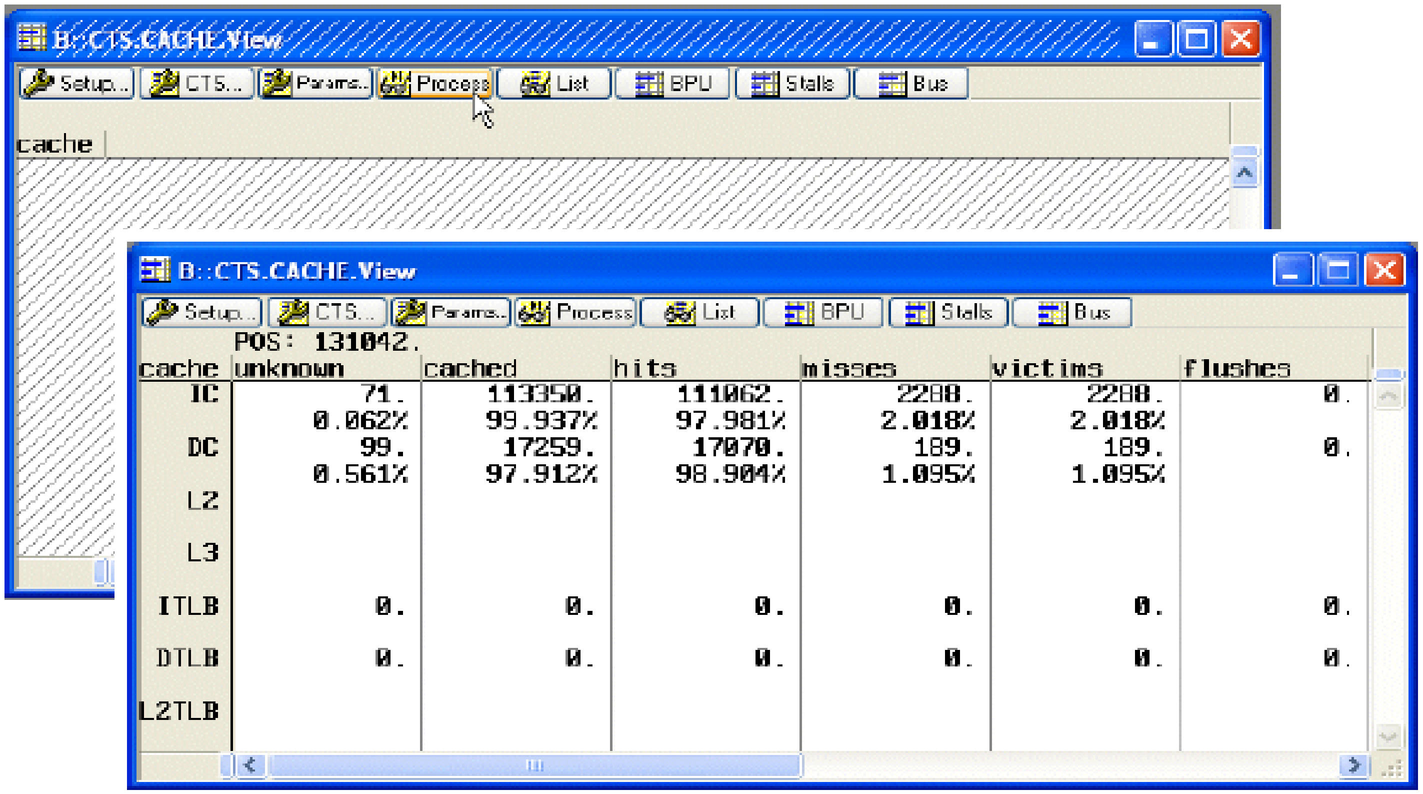
TLBArchitecture: Harvard

MMUArchitecture: NONE

Cycles: Core

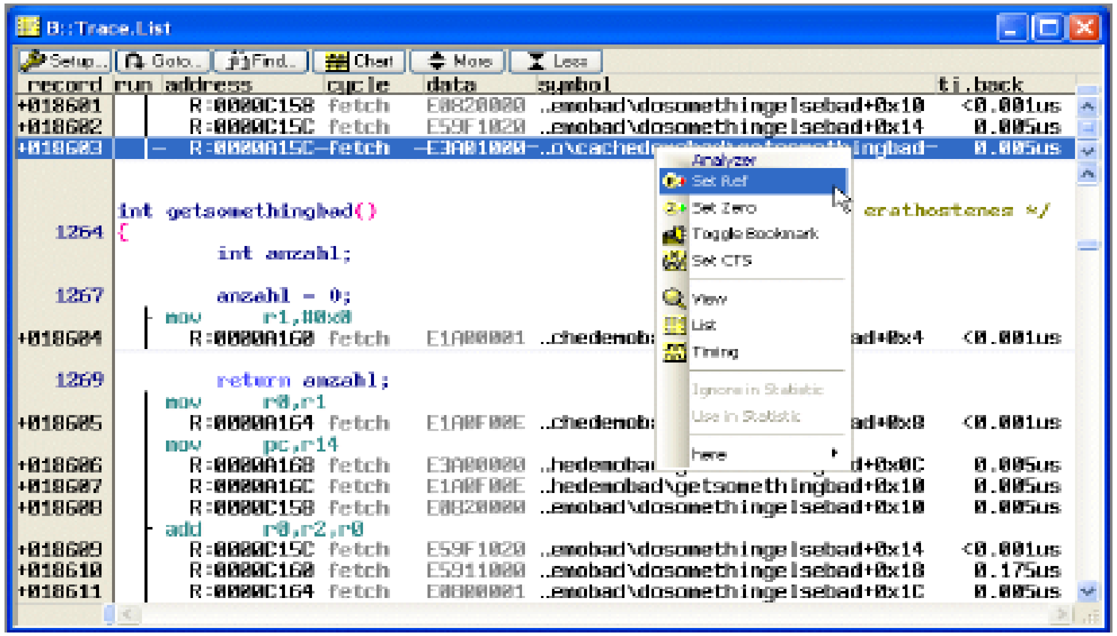
cache	SETS	WAYS	Width	Mode	Allocation	Tags	Replacement
IC	512.	2	16	WriteThrou	ReadAlloc	Logical	LRU
DC	256.	2	16	WriteThrou	ReadAlloc	Logical	LRU
L2	0	0	0.	CopyBack	ReadAlloc	Logical	Cyclic
L3	0	0	0.	CopyBack	ReadAlloc	Logical	Cyclic
ITLB			64.				Cyclic
DTLB			64.				Cyclic
L2TLB	0	0	0.				Cyclic

- 3. Push the Process button in the CTS.CACHE.VIEW window



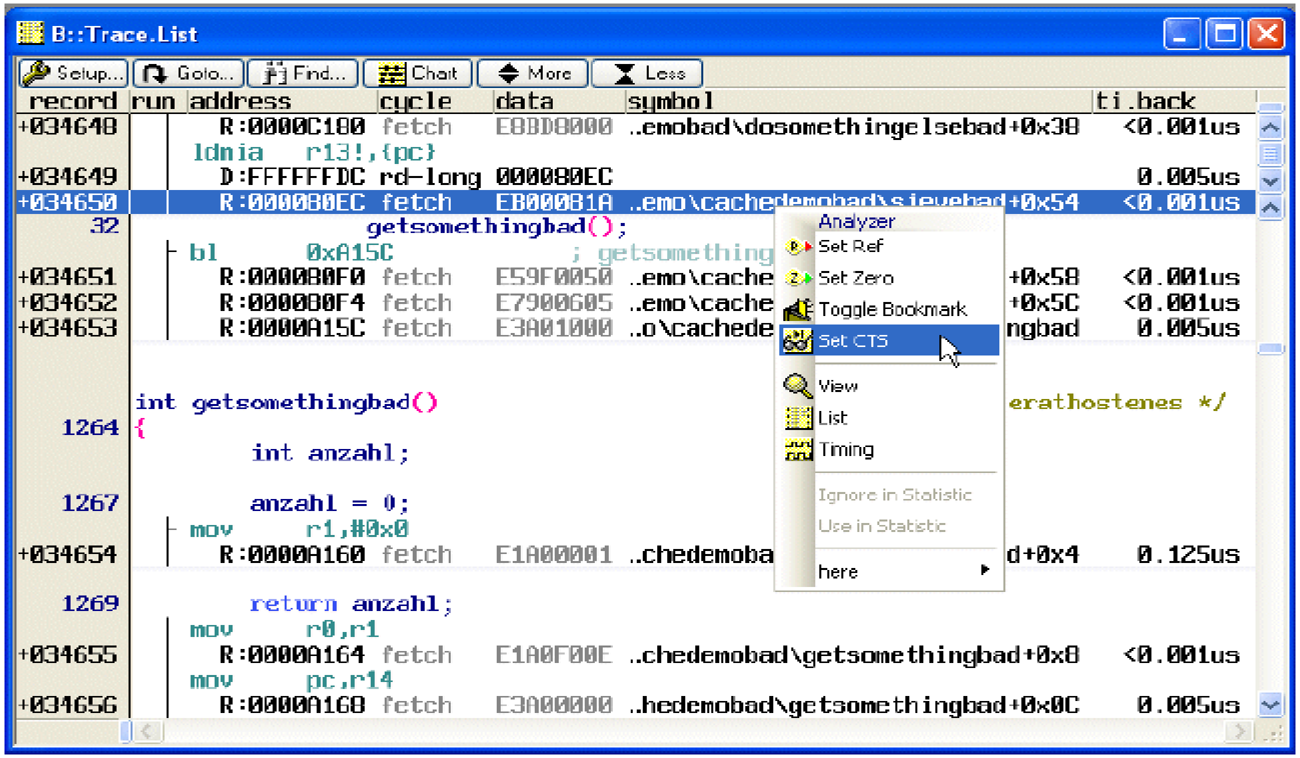
```
        POS:  131042.
cache   unknown     cached      hits        misses      victims     flushes
   IC        71.    113350.     111062.       2288.       2288.          0.
         0.062%    99.937%     97.981%      2.018%      2.018%
   DC        99.     17259.      17070.        189.        189.          0.
         0.561%    97.912%     98.904%      1.095%      1.095%
   L2
   L3
 ITLB         0.         0.          0.          0.          0.          0.
 DTLB         0.         0.          0.          0.          0.          0.
L2TLB
```

- 4. Open a trace listing and se t a reference point to the function entry:



```
record   run address       cycle  data      symbol                                      ti.back
+018601      R:0000C158   fetch  E0820000  ..emobad\dosomethingelsebad+0x10            <0.001us
+018602      R:0000C15C   fetch  E59F1020  ..emobad\dosomethingelsebad+0x14             0.005us
+018603  -   R:0000A15C   fetch  E3A01000  ..o\cachedemobad\getsomethingbad             0.005us

        int getsomethingbad()                                          erathostenes */
    1264 {
                int anzahl;

    1267        anzahl = 0;
          mov      r1,#0x0
+018604      R:0000A160   fetch  E1A00001  ..chedemob                       ad+0x4       <0.001us

    1269        return anzahl;
          mov      r0,r1
+018605      R:0000A164   fetch  E1A0F00E  ..chedemob                       ad+0x8       <0.001us
          mov      pc,r14
+018606      R:0000A168   fetch  E3A00000  ..hedemobad                      d+0x0C        0.005us
+018607      R:0000A16C   fetch  E1A0F00E  ..hedemobad\getsomethingbad+0x10             0.005us
+018608      R:0000C158   fetch  E0820000  ..emobad\dosomethingelsebad+0x10             0.005us
          add      r0,r2,r0
+018609      R:0000C15C   fetch  E59F1020  ..emobad\dosomethingelsebad+0x14            <0.001us
+018610      R:0000C160   fetch  E5911000  ..emobad\dosomethingelsebad+0x18             0.175us
+018611      R:0000C164   fetch  E0800001  ..emobad\dosomethingelsebad+0x1C             0.005us
```

- 5. Move to cursor to the end of the function and use the command SET CTS.

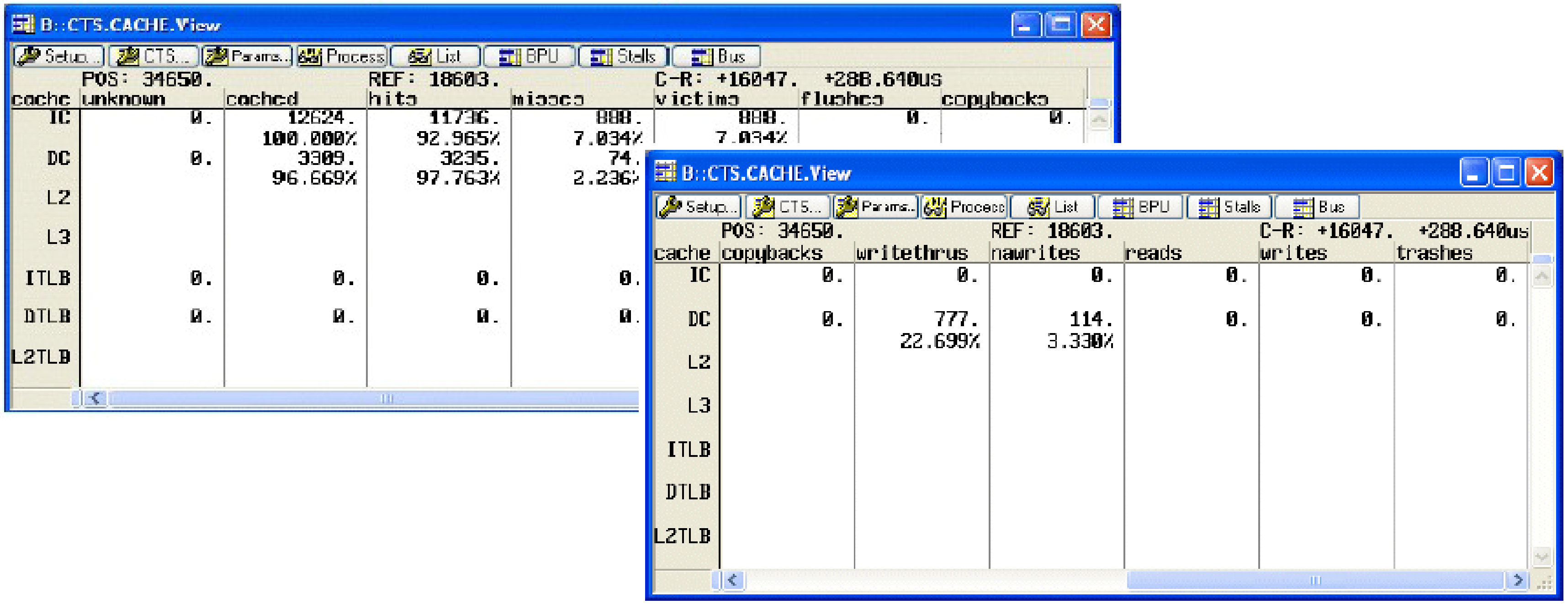


```
record run address     cycle  data     symbol                                 ti.back
+034648    R:0000C180  fetch  E8BD8000 ..emobad\dosomethingelsebad+0x38       <0.001us
       ldmia    r13!,{pc}
+034649    D:FFFFFFDC  rd-long 000080EC                                        0.005us
+034650    R:0000B0EC  fetch  EB0000B1A ..emo\cachedemobad\sievebad+0x54      <0.001us
    32                 getsomethingbad();
       bl       0xA15C              ; getsomething
+034651    R:0000B0F0  fetch  E59F0050 ..emo\cache                 +0x58      <0.001us
+034652    R:0000B0F4  fetch  E7900605 ..emo\cache                 +0x5C      <0.001us
+034653    R:0000A15C  fetch  E3A01000 ..o\cachede                 ngbad       0.005us

         int getsomethingbad()                                     erathostenes */
  1264   {
                 int anzahl;

  1267           anzahl = 0;
       mov      r1,#0x0
+034654    R:0000A160  fetch  E1A00001 ..chedemoba                 d+0x4       0.125us

  1269           return anzahl;
       mov      r0,r1
+034655    R:0000A164  fetch  E1A0F00E ..chedemobad\getsomethingbad+0x8      <0.001us
       mov      pc,r14
+034656    R:0000A168  fetch  E3A00000 ..hedemobad\getsomethingbad+0x0C       0.005us
```

The cache analysis displays the following results:

The cache analysis is performed from the trace record at the reference point (18603.) to the trace record at the current cursor position (34650.)

C-R displays the number of trace records between the current cursor position and the reference point as well as the time sper by the program between those records



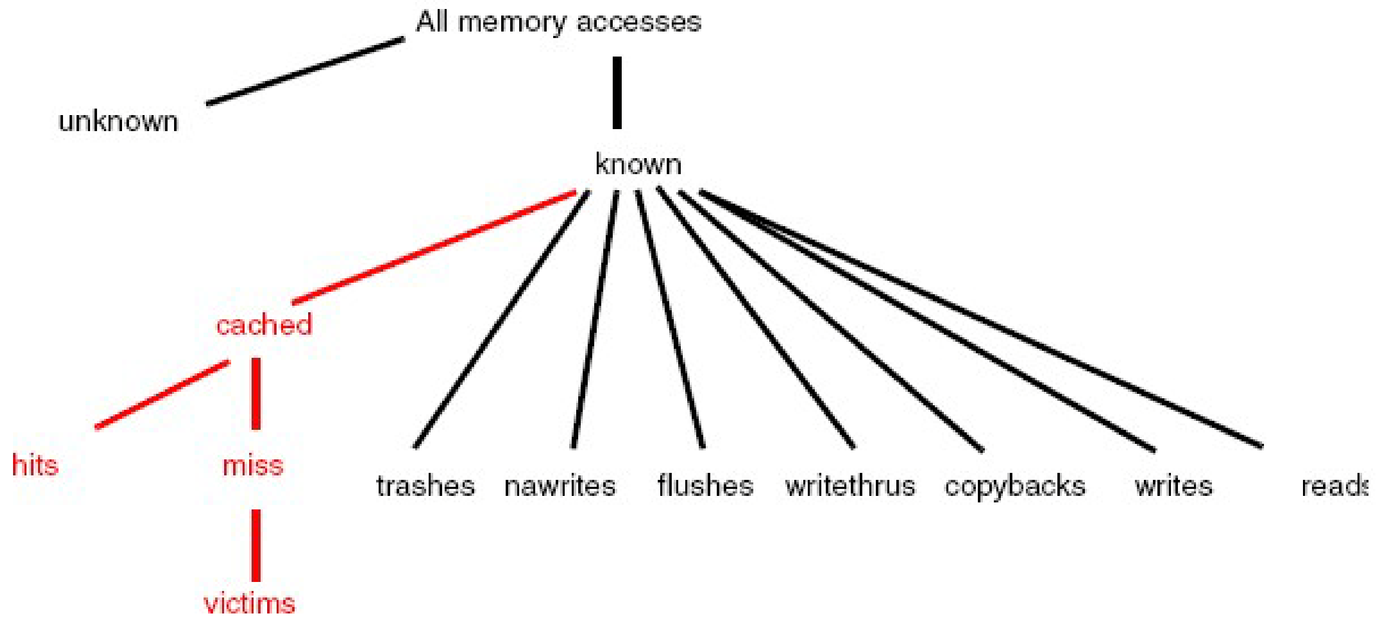
POS: 34650. REF: 18603. C-R: +16047. +288.640us

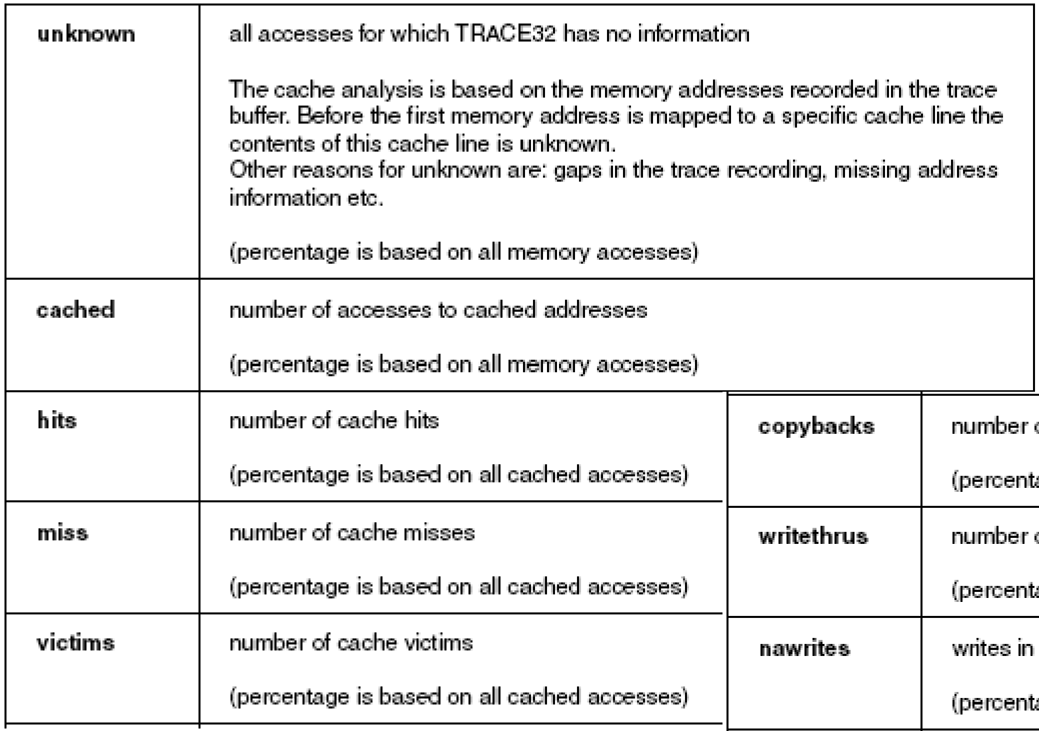
cache	unknown	cached	hits	misses	victims	flushes	copybacks
IC	0.	12624. 100.000%	11736. 92.965%	888. 7.034%	888. 7.034%	0.	0.
DC	0.	3309. 96.669%	3235. 97.763%	74. 2.236%			
L2							
L3							
ITLB	0.	0.	0.	0.			
DTLB	0.	0.	0.	0.			
L2TLB							

POS: 34650. REF: 18603. C-R: +16047. +288.640us

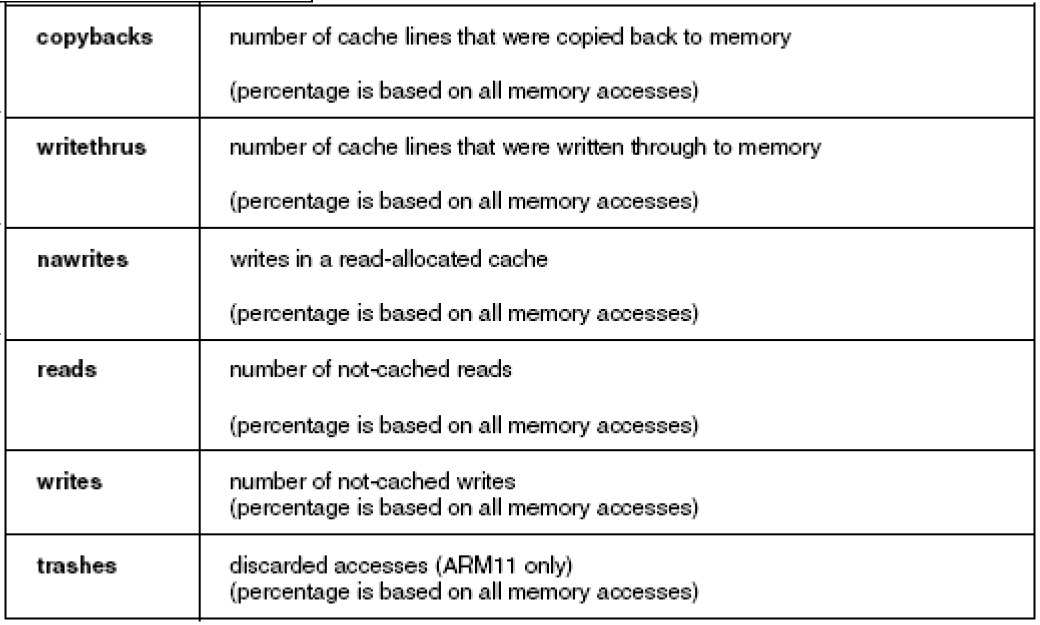
cache	copybacks	writethrus	nawrites	reads	writes	trashes
IC	0.	0.	0.	0.	0.	0.
DC	0.	777. 22.699%	114. 3.330%	0.	0.	0.
L2						
L3						
ITLB						
DTLB						
L2TLB						

Cache 性能选项解释





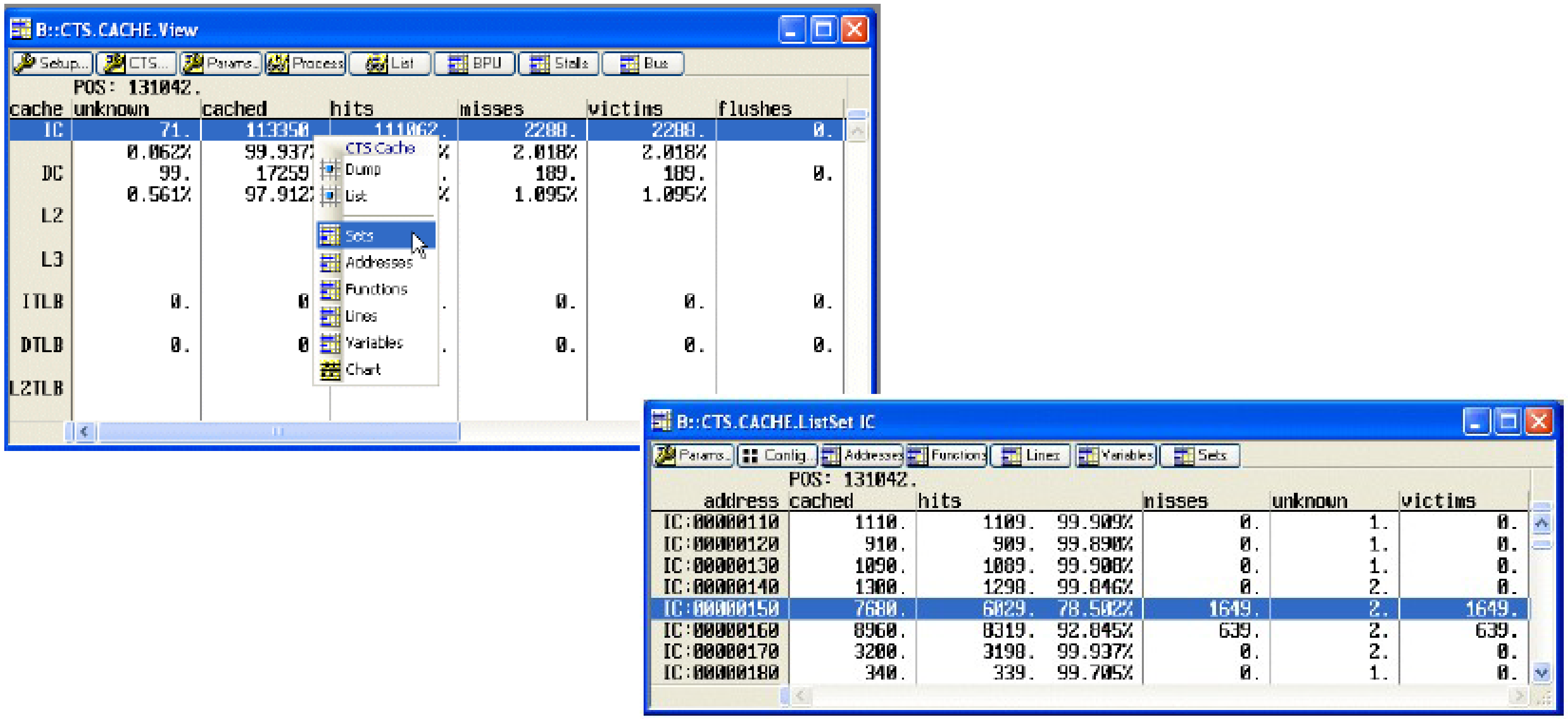
unknown	all accesses for which TRACE32 has no information The cache analysis is based on the memory addresses recorded in the trace buffer. Before the first memory address is mapped to a specific cache line the contents of this cache line is unknown. Other reasons for unknown are: gaps in the trace recording, missing address information etc. (percentage is based on all memory accesses)
cached	number of accesses to cached addresses (percentage is based on all memory accesses)
hits	number of cache hits (percentage is based on all cached accesses)
miss	number of cache misses (percentage is based on all cached accesses)
victims	number of cache victims (percentage is based on all cached accesses)



copybacks	number of cache lines that were copied back to memory (percentage is based on all memory accesses)
writethrus	number of cache lines that were written through to memory (percentage is based on all memory accesses)
nawrites	writes in a read-allocated cache (percentage is based on all memory accesses)
reads	number of not-cached reads (percentage is based on all memory accesses)
writes	number of not-cached writes (percentage is based on all memory accesses)
trashes	discarded accesses (ARM11 only) (percentage is based on all memory accesses)

Analysis for single Cache Set

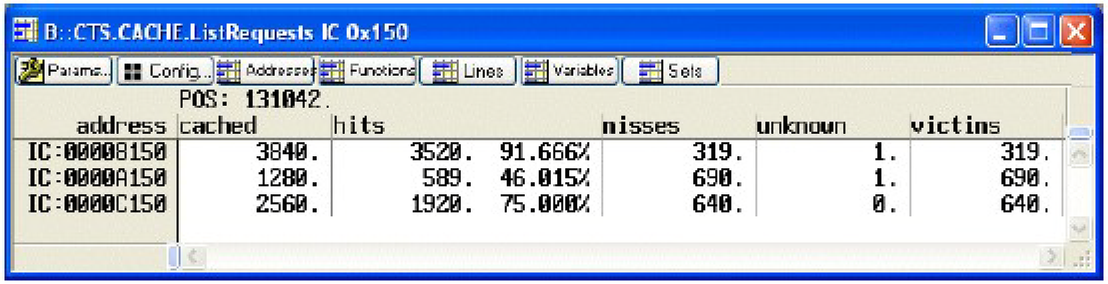
- Which cache sets have a particular high cache victim rate?(too many memory addresses are mapped to these sets)
- Are there any cache sets not being or hardly being used?(sets that are suitable for address relocated)



```
      POS: 131042.
cache unknown     cached      hits          misses       victims      flushes
   IC        71.      113350      111062.        2288.        2288.          0.
          0.062%      99.937                    2.018%       2.018%
   DC        99.      17259                       189.         189.          0.
          0.561%      97.912                    1.095%       1.095%
   L2
   L3
 ITLB         0.          0                         0.           0.          0.
 DTLB         0.          0                         0.           0.          0.
L2TLB
```

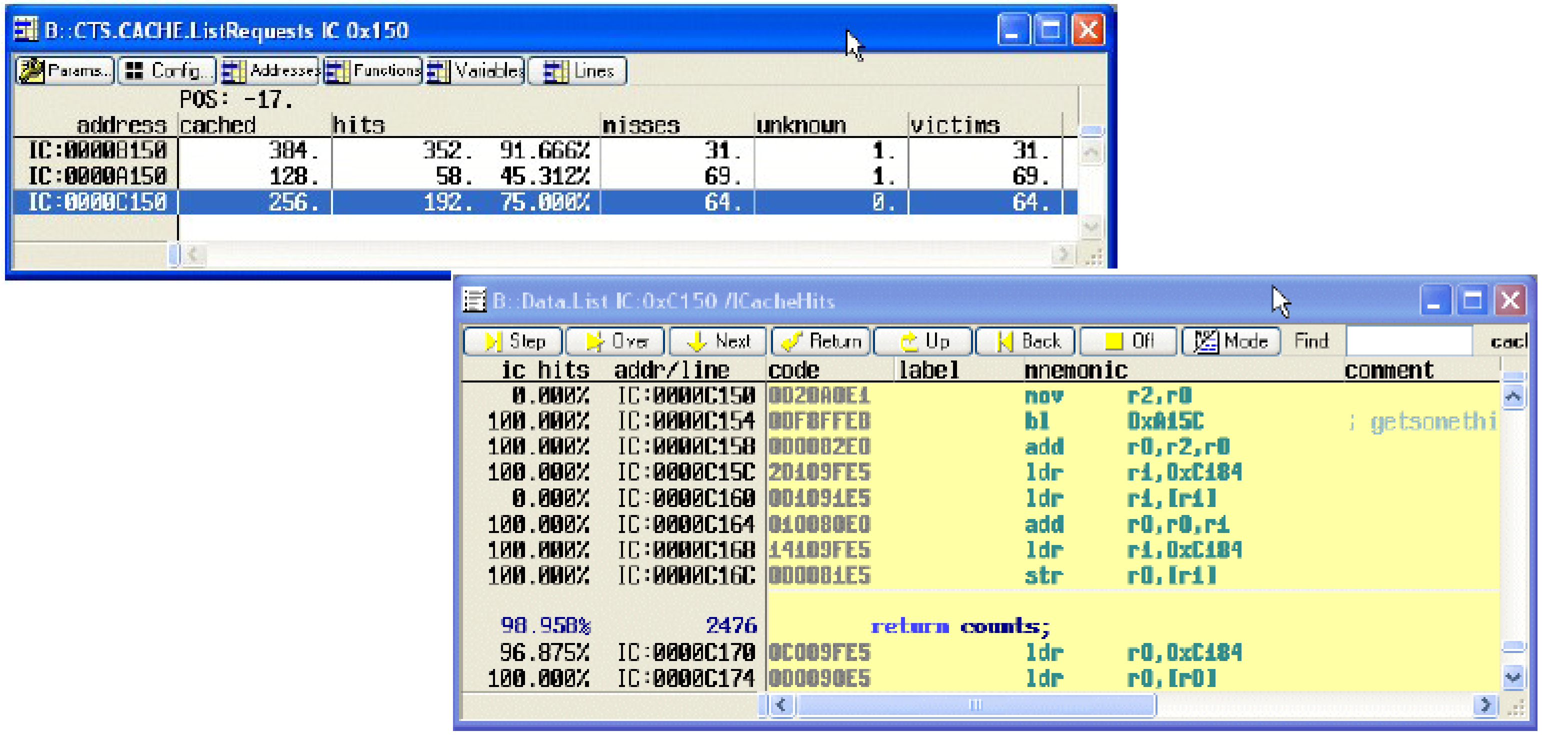
```
                POS: 131042.
        address cached     hits                misses       unknown      victims
IC:00000110       1110.      1109.   99.909%          0.           1.          0.
IC:00000120        910.       909.   99.890%          0.           1.          0.
IC:00000130       1090.      1089.   99.908%          0.           1.          0.
IC:00000140       1300.      1298.   99.846%          0.           2.          0.
IC:00000150       7680.      6029.   78.502%       1649.           2.       1649.
IC:00000160       8960.      8319.   92.845%        639.           2.        639.
IC:00000170       3200.      3198.   99.937%          0.           2.          0.
IC:00000180        340.       339.   99.705%          0.           1.          0.
```

- Which memory address compete for heavily used cache set?(code/data that has to be relocated)
- Double click the address..



```
             POS: 131042.
     address cached     hits                     misses     unknown     victims
IC:00008150     3840.      3520.  91.666%         319.         1.        319.
IC:0000A150     1280.       589.  46.015%         690.         1.        690.
IC:0000C150     2560.      1920.  75.000%         640.         0.        640.
```

- View the memory contents for a selected address by double click.


B::CTS.CACHE.ListRequests IC 0x150
Params... Config... Addresses Functions Variables Lines
B::Data.List IC:0xC150 /ICacheHits
Step Over Next Return Up Back Off Mode Find

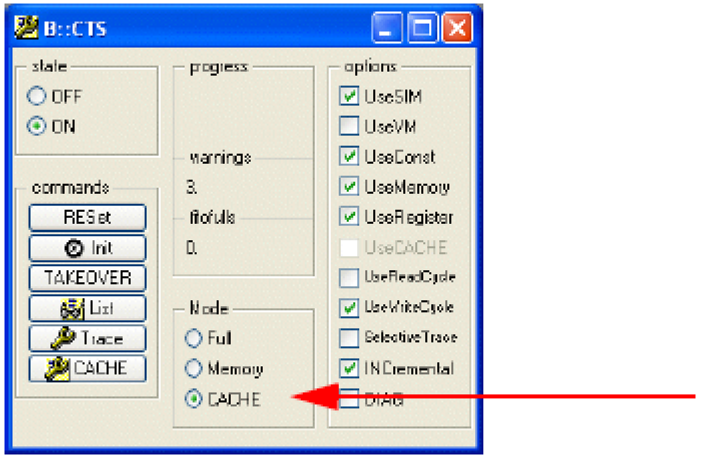
```
              POS: -17.
       address cached        hits                  misses     unknown    victims
   IC:00008150      384.        352.   91.666%        31.          1.        31.
   IC:0000A150      128.         58.   45.312%        69.          1.        69.
   IC:0000C150      256.        192.   75.000%        64.          0.        64.
```

```
   ic hits  addr/line    code      label    mnemonic            comment
    0.000%  IC:0000C150  0020A0E1           mov     r2,r0
  100.000%  IC:0000C154  0DF8FFEB           bl      0xA15C      ; getsomethi
  100.000%  IC:0000C158  000082E0           add     r0,r2,r0
  100.000%  IC:0000C15C  20109FE5           ldr     r1,0xC184
    0.000%  IC:0000C160  001091E5           ldr     r1,[r1]
  100.000%  IC:0000C164  010080E0           add     r0,r0,r1
  100.000%  IC:0000C168  14109FE5           ldr     r1,0xC184
  100.000%  IC:0000C16C  000081E5           str     r0,[r1]

   98.958%         2476        return counts;
   96.875%  IC:0000C170  0C009FE5           ldr     r0,0xC184
  100.000%  IC:0000C174  000090E5           ldr     r0,[r0]
```

Calculation Details

See how much number of cache hits\misses\ unknown for each assemble line process as follows:

- 1. Select CTS mode CACHE in the CTS window:



B::CTS

state
- OFF
- [✓ ON]

commands
- RESet
- Init
- TAKEOVER
- List
- Trace
- CACHE

progress

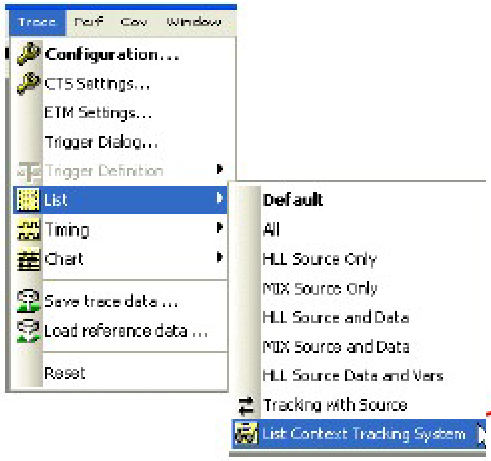
warnings
3.

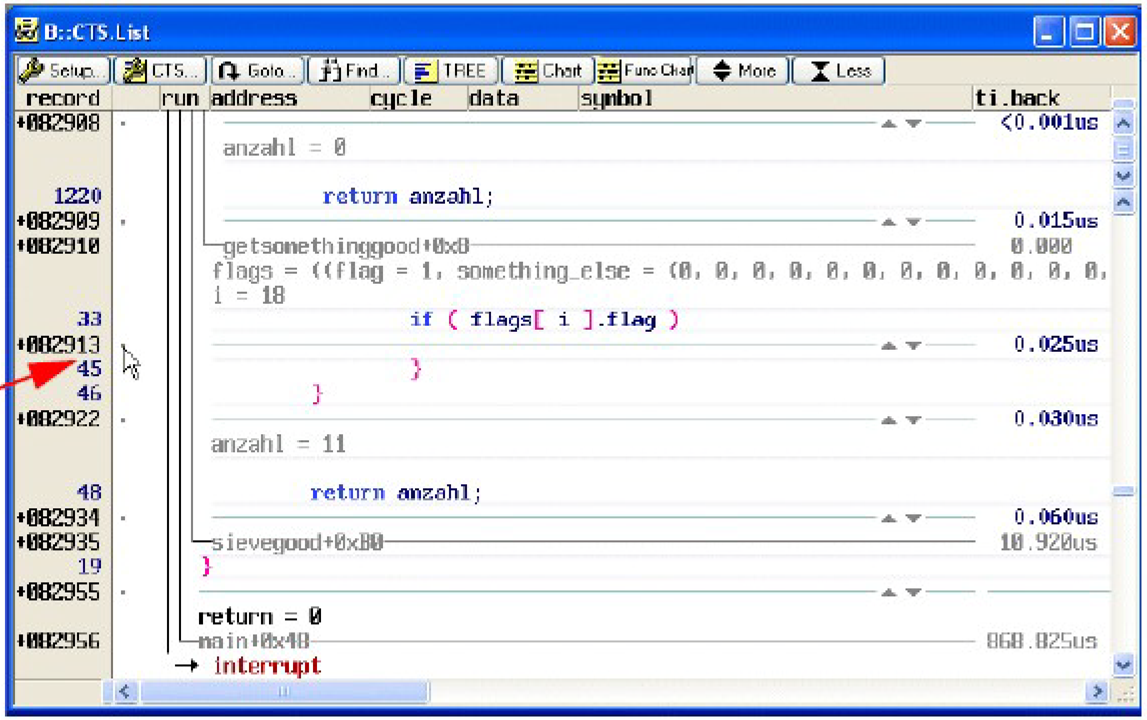
pitfalls
0.

Mode
- Full
- Memory
- [✓ CACHE]

options
- [✓] UseSIM
- [ ] UseVM
- [✓] UseConst
- [✓] UseMemory
- [✓] UseRegister
- [ ] UseCACHE
- [ ] UseReadCycle
- [✓] UseWriteCycle
- [ ] SelectiveTrace
- [✓] INCremental
- [ ] DIAG

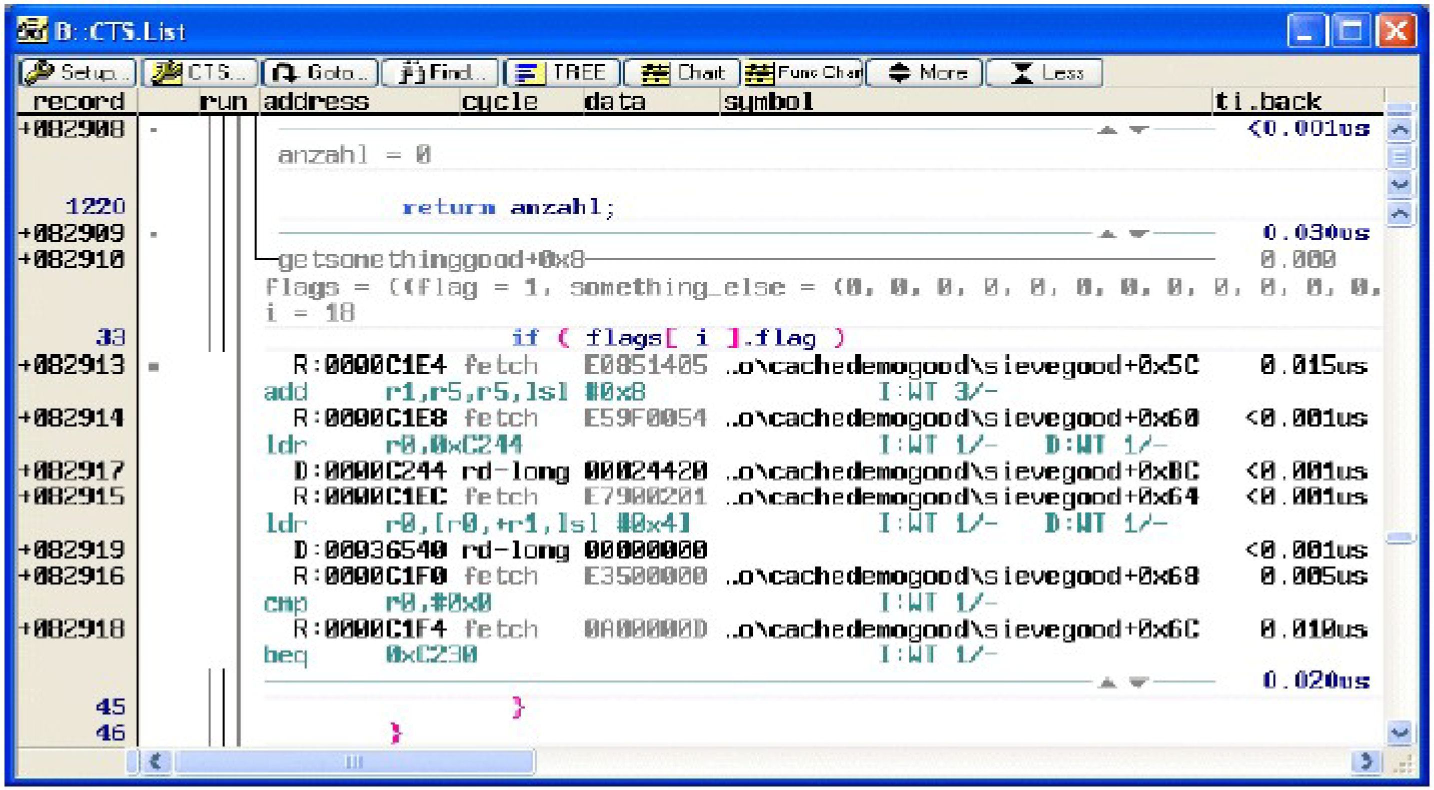
- 2. Open a CTS.LIST Window





```
record       run address       cycle   data      symbol                     ti.back
+082908  .                                                                  <0.001us
              anzahl = 0
    1220                     return anzahl;
+082909  .                                                                   0.015us
+082910       getsomethinggood+0x8                                           0.000
           flags = ((flag = 1, something_else = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           i = 18
      33                            if ( flags[ i ].flag )
+082913  .                                                                   0.025us
      45                            }
      46                     }
+082922  .                                                                   0.030us
           anzahl = 11
      48                     return anzahl;
+082934  .                                                                   0.060us
+082935    sievegood+0xB0                                                   10.920us
      19   }
+082955  .
           return = 0
+082956    main+0x48                                                       868.825us
         → interrupt
```

- 3. Click to the small dot beside a hll line



```
record    run address        cycle   data     symbol                                  ti.back
+082908 .                                                                             <0.001us
            anzahl = 0

1220                 return anzahl;
+082909 .                                                                             0.030us
+082910     getsomethinggood+0x8                                                      0.000
           flags = ((flag = 1, something_else = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           i = 18
33                         if ( flags[ i ].flag )
+082913 .      R:0000C1E4 fetch   E0851405 ..o\cachedemogood\sievegood+0x5C    0.015us
           add       r1,r5,r5,lsl #0x8                    I:WT 3/-
+082914        R:0000C1E8 fetch   E59F0054 ..o\cachedemogood\sievegood+0x60   <0.001us
           ldr       r0,0xC244                            I:WT 1/-   D:WT 1/-
+082917        D:0000C244 rd-long 00024420 ..o\cachedemogood\sievegood+0xBC   <0.001us
+082915        R:0000C1EC fetch   E7900201 ..o\cachedemogood\sievegood+0x64   <0.001us
           ldr       r0,[r0,+r1,lsl #0x4]                 I:WT 1/-   D:WT 1/-
+082919        D:00036540 rd-long 00000000                                    <0.001us
+082916        R:0000C1F0 fetch   E3500000 ..o\cachedemogood\sievegood+0x68    0.005us
           cmp       r0,#0x0                              I:WT 1/-
+082918        R:0000C1F4 fetch   0A00000D ..o\cachedemogood\sievegood+0x6C    0.010us
           beq       0xC230                               I:WT 1/-
                                                                               0.020us
45                         }
46                     }
```

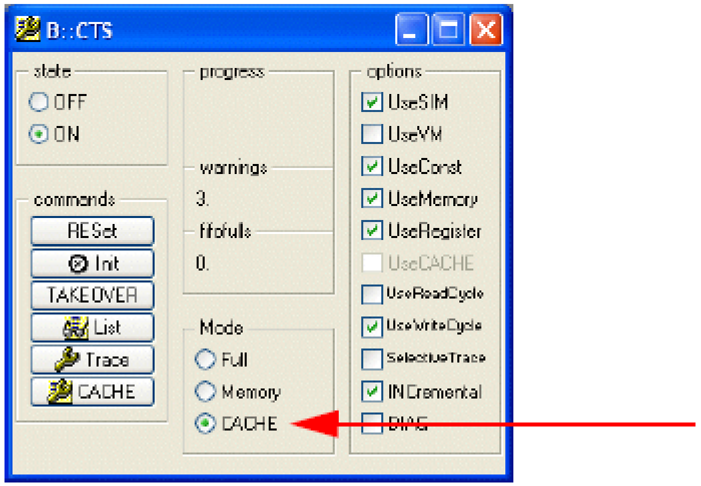
I:*<mode> <hits>/<misses>*	Instruction cache hits and misses
I:*<mode> <number>?*	Number of unknown accesses to the instruction cache
I: NC*<number>*	Number of accesses to not cached instruction addresses
D:*<mode> <hits>/<misses>*	Data cache hits and misses
D:*<mode> <number>?*	Number of unknown accesses to the data cache
D: NC*<number>*	Number of accesses to not cached data addresses

Mode	
WT	Write Through
CB	Copy Back

Trace-based debugging with cache display

If you perform trace-based debugging you can also watch the changes in the caches

- 1. select CTS mode CACHE in CTS window



State: OFF [✓ ON]

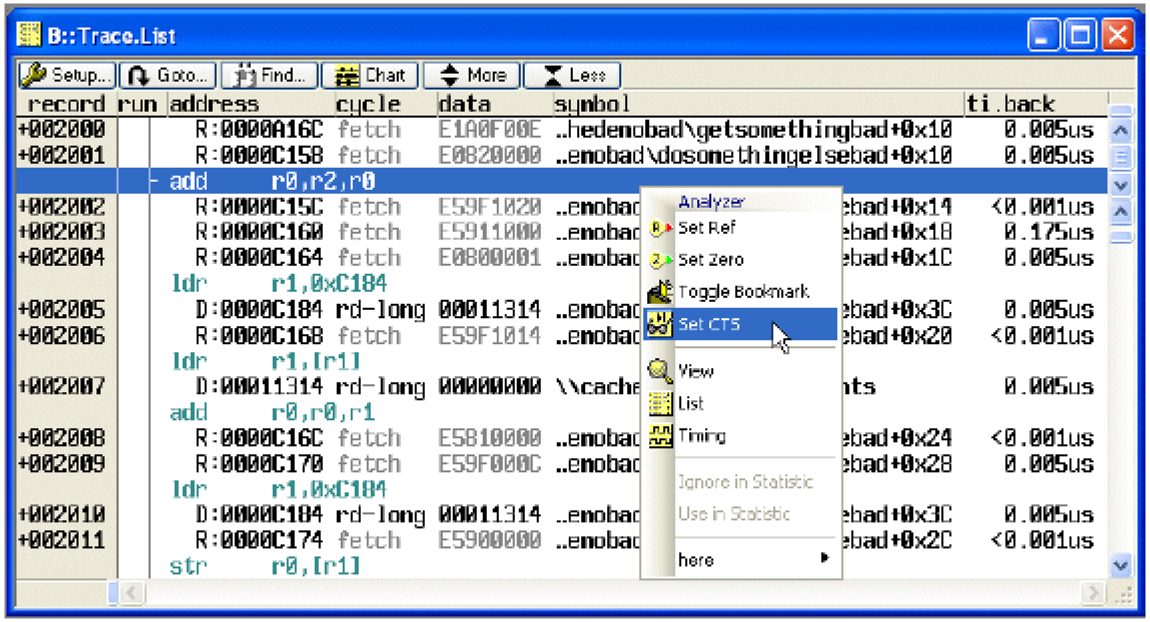
Options: [✓ UseSIM] UseVM [✓ UseConst] [✓ UseMemory] [✓ UseRegister] UseCACHE UseReadCycle [✓ UseWriteCycle] SelectiveTrace [✓ INCremental] DIAG

warnings: 3.

fifofulls: 0.

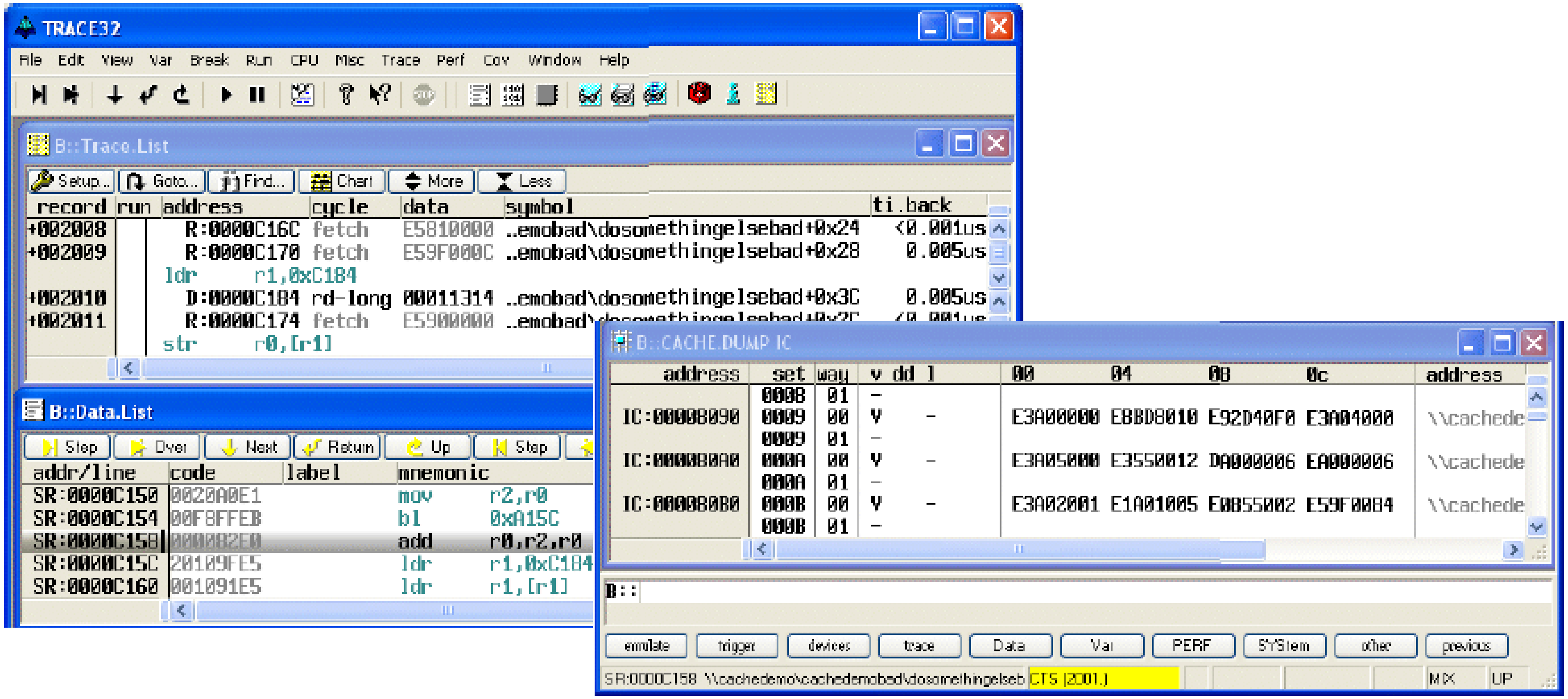
Mode: Full Memory [✓ CACHE]

2.Select the start point for trace-based debugging



```
record  run address    cycle   data     symbol                              ti.back
+002000     R:0000A16C fetch   E1A0F00E ..hedenobad\getsomethingbad+0x10    0.005us
+002001     R:0000C158 fetch   E0820000 ..enobad\dosomethingelsebad+0x10    0.005us
          add       r0,r2,r0
+002002     R:0000C15C fetch   E59F1020 ..enobac            ebad+0x14      <0.001us
+002003     R:0000C160 fetch   E5911000 ..enobac            ebad+0x18       0.175us
+002004     R:0000C164 fetch   E0800001 ..enobac            ebad+0x1C       0.005us
          ldr       r1,0xC184
+002005     D:0000C184 rd-long 00011314 ..enobac            ebad+0x3C       0.005us
+002006     R:0000C168 fetch   E59F1014 ..enobac            ebad+0x20      <0.001us
          ldr       r1,[r1]
+002007     D:00011314 rd-long 00000000 \\cache             nts             0.005us
          add       r0,r0,r1
+002008     R:0000C16C fetch   E5810000 ..enobac            ebad+0x24      <0.001us
+002009     R:0000C170 fetch   E59F000C ..enobac            ebad+0x28       0.005us
          ldr       r1,0xC184
+002010     D:0000C184 rd-long 00011314 ..enobac            ebad+0x3C       0.005us
+002011     R:0000C174 fetch   E5900000 ..enobac            ebad+0x2C      <0.001us
          str       r0,[r1]
```

- 3. Open a CACHE.DUMP window to watch the changes in the cache while re-debugging the trace contents.


TRACE32
File Edit View Var Break Run CPU Misc Trace Perf Cov Window Help
B::Trace.List
Setup... Goto... Find... Chart More Less
B::Data.List
Step Over Next Return Up Step
B::CACHE.DUMP IC
emulate trigger devices trace Data Var PERF SYStem other previous

```
record run address       cycle   data     symbol                              ti.back
+002008      R:0000C16C  fetch   E5810000 ..emobad\dosomethingelsebad+0x24    <0.001us
+002009      R:0000C170  fetch   E59F000C ..emobad\dosomethingelsebad+0x28     0.005us
           ldr        r1,0xC184
+002010      D:0000C184  rd-long 00011314 ..emobad\dosomethingelsebad+0x3C     0.005us
+002011      R:0000C174  fetch   E5900000 ..emobad\
           str        r0,[r1]
```

```
addr/line      code      label   mnemonic
SR:0000C150    0020A0E1          mov     r2,r0
SR:0000C154    00F8FFEB          bl      0xA15C
SR:0000C158    000082E0          add     r0,r2,r0
SR:0000C15C    20109FE5          ldr     r1,0xC184
SR:0000C160    001091E5          ldr     r1,[r1]
```

```
      address   set  way  v dd ]    00       04       08       0c       address
                0008 01   -
IC:0000B090     0009 00   V   -     E3A00000 E8BD8010 E92D40F0 E3A04000 \\cachede
                0009 01   -
IC:0000B0A0     000A 00   V   -     E3A05000 E3550012 DA000006 EA000006 \\cachede
                000A 01   -
IC:0000B0B0     000B 00   V   -     E3A02001 E1A01005 E0855002 E59F0084 \\cachede
                000B 01   -
B::
SR:0000C158 \\cachedemo\cachedemobad\dosomethingelseb CTS [2001.] MIX UP
```

Cache analysis whit an trace32 instruction set simulator

- Export required data from PowerTrace

```
Trace.SAVE cache                               ; save the contents of the
                                               ; trace buffer to a file

Data.SAVE.S3record mmu A:0x1000++0xfff         ; Save MMU translation table
                                               ; to file
```

- Check the value of the MMU control register, they need to be set the same values in the TRACE32 instruction set simulator

Setup the TRACE32 instruction set simulator for the cache analysis

- 1. Select the CPU and activate the simulator

```
SYStem.Down

SYStem.CPU ARM925T

SYStem.Up
```

- 2. Load the MMU translation table

```
Data.LOAD.S3record mmu
```

- 3. Configure the MMU control register

```
Data.Set C15:1 0x127f          ; example for ARM9 here

Data.Set C15:2 0xa000000

Data.Set C15:3 0x55555555
```

- 4. Load trace contents from file

```
Trace.LOAD cache

Trace.List /FILE              ; display trace contents loaded from
                              ; the file
```

- 5. Configure CTS

```
CTS.Mode CACHE                ; switch on the cache analysis within
                              ; CTS

CTS.UseMemory OFF             ; current contents of the target memory
                              ; can not be used for cache analysis

CTS.UseRegister OFF           ; current contents of the CPU registers
                              ; can not be used for cache analysis
```

- 6. Configure the cache structure ,if this is not automatically done by selecting the CPU

```
CTS.CACHE.WAYS 4.

CTS.CACHE.SETS IC 512.

CTS.CACHE.SETS DC 128.

…
```

- 7. Process the cache analysis

```
CTS.PROCESS /FILE
```

Q & A

Thanks for your attendance!

Sales:sales_cn@lauterbach

Support:support_cn@lauterbach

FAQ: http://www.lauterbach.com/faq.html