COLDSTART

krok by krok
tick by tick pray look
at isch code. na pryktadzie diagnostyki tut.
test cpu 8030

```
login: root
Password:
Last login: Tue Feb  9 10:30:11 from convext
Feb  9 10:35:39 trainc240 login: ROOT LOGIN console
root@trainc240# w
 10:35am  up  1:59,  1 user,  load average: 0.07, 0.05, 0.03
User     tty        login@  idle   JCPU   PCPU  what
root     console  10:35am                     w
root@trainc240# shutdown -h now
Shutdown at 10:35 (in 0 minutes) [pid 508]


        *** FINAL System shutdown message from root@trainc240 ***

System going down IMMEDIATELY


System shutdown time has arrived
root@trainc240# Feb  9 10:35:52 trainc240 syslogd: exiting on signal 15
[CPU01@10:36:05] syncing disks...
[CPU01@10:36:05] done
[CPU01@10:36:05] halting in tight loop; type "sysreset" to halt
/mnt/os/boot: 119 Terminated

Cleaning up SPU processes...
(spu)> sysreset -l2
sysreset:   revision 5.1 (Mon Apr 27 11:43:53 1992)
(spu)> mminit  zerowanie kompletne pamięci
mminit:   revision 5.1 (Mon Apr 27 11:43:52 1992)
mminit:   using PCM from /mnt/boot_db
mminit:   initializing PI, CU, and SP PCMs:   0:00:08

Main memory size:  134217728 bytes - 131072 K - 128 Meg

pair  allocated 16 meg PCM blocks, from a system perspective
 0      0   1   2   3
 1     64  65  66  67
 2
 3

mminit:   interleave set to 16-way
mminit:   using CPU 0 to initialize memory:   0:00:09
mminit:   memory initialization complete
(spu)> dshell
```

oglądanie w pc – jcpureg
isch heap
glan - hang
mmq - fu_mmqueue
orbiter deb gate - dump-ring
memory - mm, mem-scr
output xbar - dump-rd-of
dfw - asp func

```
CONVEX DIAGNOSTIC SHELL

: test cpu4030
cpu4030.t:    unable to deterermine software revision
```

Test 'cpu4030.t' Tue Feb 9 10:38:42 1993

ENTER TEST PARAMETERS

[]	Encloses allowed input ranges or values
()	Encloses the default value
^	Returns to the previous prompt
:nn	Returns to the prompt # nn
:	Returns to the first unsatisfied prompt
:?	Reviews previous entries

1: Run default switches? [y,n] (y) -> CR

TEST PARAMETER SUMMARY

Run default switches? : y

Loading and initializing test ... 0:03:01

Current Memory Allocation

File No.	Physical Address	Pid	File Name	Logical Offset
1	00200000-00207fff	0	p0r0_4030	00000000
2	00208000-0023bfff	0	cpu4030.rnn	00012000
3	0023c000-0023dfff	0	p0rN_4030	20000000
4	0023e000-0023ffff	0	p0rN_4030	40000000
5	00240000-00241fff	0	p0rN_4030	60000000
6	00242000-00243fff	0	p0rN_4030	80000000
7	00244000-00246fff	0	pccarry_4030	0000f000
8	00247000-00249fff	0	pccarry_4030	000ff000
9	0024a000-0024cfff	0	pccarry_4030	00fff000
10	0024d000-0024ffff	0	pccarry_4030	0ffff000
11	00250000-00250fff	0	wrapu_4030	1ffff000
12	00251000-00251fff	0	wrapu_4030	3ffff000
13	00252000-00252fff	0	wrapu_4030	5ffff000
14	00253000-00253fff	0	wrapu_4030	7ffff000
15	00254000-00254fff	0	wrapu_4030	9ffff000
16	00255000-00255fff	0	wrapu_4030	bffff000
17	00256000-00256fff	0	wrapu_4030	dffff000
18	00257000-00257fff	0	wrapu_4030	fffff000
19	00258000-00258fff	0	wrapl_4030	a0000000
20	00259000-00259fff	0	wrapl_4030	c0000000
21	0025a000-0025afff	0	wrapl_4030	e0000000
22	0025b000-00262fff	1	p0r0_4030	00000000
23	00263000-00296fff	1	cpu4030.rnn	00012000
24	00297000-00298fff	1	p0rN_4030	20000000
25	00299000-0029afff	1	p0rN_4030	40000000
26	0029b000-0029cfff	1	p0rN_4030	60000000
27	0029d000-0029efff	1	p0rN_4030	80000000
28	0029f000-002a1fff	1	pccarry_4030	0000f000
29	002a2000-002a4fff	1	pccarry_4030	000ff000
30	002a5000-002a7fff	1	pccarry_4030	00fff000
31	002a8000-002aafff	1	pccarry_4030	0ffff000
32	002ab000-002abfff	1	wrapu_4030	1ffff000
33	002ac000-002acfff	1	wrapu_4030	3ffff000
34	002ad000-002adfff	1	wrapu_4030	5ffff000
35	002ae000-002aefff	1	wrapu_4030	7ffff000
36	002af000-002affff	1	wrapu_4030	9ffff000
37	002b0000-002b0fff	1	wrapu_4030	bffff000

```
    38   002b1000-002b1fff    1 wrapu_4030                    dffff000
    39   002b2000-002b2fff    1 wrapu_4030                    fffff000
    40   002b3000-002b3fff    1 wrapl_4030                    a0000000
    41   002b4000-002b4fff    1 wrapl_4030                    c0000000
    42   002b5000-002b5fff    1 wrapl_4030                    e0000000
   -----  23fd6000-23fe9fff    1 pte2                         NA
   -----  23fea000-23feafff    1 pte1                         NA
   -----  23feb000-23ffefff    0 pte2                         NA
   -----  23fff000-23ffffff    0 pte1                         NA
        Subtest    1  CPU:0   0:00:02   passed
                      CPU:1   0:00:00   passed
        Subte  0:00:01

^C MENU

Enter:  0  to continue test
        1  to abort test
        2  to abort subtest
        3  to abort and pause at end of current subtest
> 1

Test 'cpu4030.t' aborted
: q
(spu)> mm   <- rozkaz z spu - reszta opisana w Maintenance boot console
mm:   revision 5.1 (Mon Apr 27 11:43:52 1992)
mm:address mode = PHYSICAL
mm(00,00):s log
  CIR  TID
   0    0                  Current sdrs:

sdr[0]=23fffe10 sdr[1]=23fffc10 sdr[2]=23fffa10 sdr[3]=23fff810
sdr[4]=23fff610 sdr[5]=23fff410 sdr[6]=23fff210 sdr[7]=23fff010
mm(00,00):!cpureg   oglądamy zawartość rejestrów cpu.

Register dump for cpu: 0
a0: 00001000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 00012140  psw:  000000c0
a2: 00000005  s2: 00000000 00000001  t2: 00000000  ipc:       000
a3: 00000003  s3: ffffffff ffffffff  t3: 00000002  ccr:    680000
a4: 00000004  s4: 00000000 00000000  t4: 00000019  cir:     0  tid: 00
a5: 00000002  s5: 00000000 00000000  t5: 00012000   vl:   00   vs: 00000000
a6: 00000001  s6: a9a082a8 00000000  t6: 0ad6bad7 vm_u:  00000000 00000000
a7: 00000007  s7: 00000000 00000006  t7: 0ad4bad5 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

Register dump for cpu: 1
a0: 0b2a8eb4  s0: 00000000 00000000  t0: 003ffff8   pc:  00000000
a1: 00000000  s1: 00000000 ffffffff  t1: 003fe000  psw:  000000c0
a2: 00000005  s2: 00080a24 00000053  t2: 00000ff8  ipc:       000
a3: 00000003  s3: 00000000 00000021  t3: 00000000  ccr:    000000
a4: 00000004  s4: 00000000 000000fc  t4: 0000001b  cir:     0  tid: 00
a5: 00000002  s5: 00000000 00000001  t5: 00012000   vl:   00   vs: 00000000
a6: 00000001  s6: 00000000 8000c86e  t6: 0ad6bad7 vm_u:  00000000 00000000
a7: 00000007  s7: 00000000 00000000  t7: 0ad4bad5 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0
mm(00,00):c 1
zmiana CR 0 na CR 1   tam SDR
                    Current sdrs:

sdr[0]=23feae10 sdr[1]=23feac10 sdr[2]=23feaa10 sdr[3]=23fea810
sdr[4]=23fea610 sdr[5]=23fea410 sdr[6]=23fea210 sdr[7]=23fea010
mm(01,00):c 0
                    Current sdrs:

sdr[0]=23fffe10 sdr[1]=23fffc10 sdr[2]=23fffa10 sdr[3]=23fff810
sdr[4]=23fff610 sdr[5]=23fff410 sdr[6]=23fff210 sdr[7]=23fff010
mm(00,00):i 12000,15      i - include wyświetla od tego adresu 15 instr.
0x00012000      ld.w      #0x55555555,a1
```

```
0x000120d6      ld.w    #0xaaaaaaaa,a2
0x000120dc      ld.w    #0xffff,a3
0x000120e0      ld.w    #0xeeeeeeee,a4
0x000120e6      ld.w    #0x77777777,a5
0x000120ec      ld.w    #0x99999999,a6
0x000120f2      ld.w    #0x66666666,a7
0x000120f8      ld.w    #0xaaaaaaaa,a1
0x000120fe      ld.w    #0x55555555,a2
0x00012104      ld.w    #0x0,a3
0x00012106      ld.w    #0x11111111,a4
0x0001210c      ld.w    #0x88888888,a5
0x00012112      ld.w    #0x66666666,a6
0x00012118      ld.w    #0x99999999,a7
0x0001211e      ld.h    #0x0,a1
0x00012120      ld.h    #0x1,a2
0x00012122      ld.h    #0x2,a3
0x00012124      ld.h    #0x3,a4
0x00012126      ld.h    #0x4,a5
0x00012128      ld.h    #0x5,a6
0x0001212a      ld.h    #0x6,a7
```

11 - opcode
C1 - register select 5555 - data

$40_{16} = 64_{10}$

```
mm(00,00):d 120d0,40    wyświetl 40 bytów from adres 120d0
000120d0 11 c1 55 55 55 55 11 c2 aa aa aa aa 11 83 ff ff   ..UUUU.. ........
000120e0 11 c4 ee ee ee ee 11 c5 77 77 77 77 11 c6 99 99   ........ wwww....
000120f0 99 99 11 c7 66 66 66 66 11 c1 aa aa aa aa 11 c2   ....ffff ........
00012100 55 55 55 55 44 c3 11 c4 11 11 11 11 11 c5 88 88   UUUUD... ........
mm(00,00):q
(spu)> cd /hw/cputest
(spu)> cpureg

Register dump for cpu: 0
a0: 00001000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 00012140  psw:  000000c0
a2: 00000005  s2: 00000000 00000001  t2: 00000000  ipc:       000
a3: 00000003  s3: ffffffff ffffffff  t3: 00000002  ccr:    680000
a4: 00000004  s4: 00000000 00000000  t4: 00000019  cir:     0  tid: 00
a5: 00000002  s5: 00000000 00000000  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000001  s6: a9a082a8 00000000  t6: 0ad6bad7 vm_u:  00000000 00000000
a7: 00000007  s7: 00000000 00000006  t7: 0ad4bad5 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

Register dump for cpu: 1
a0: 0b2a8eb4  s0: 00000000 00000000  t0: 003ffff8   pc:  00000000
a1: 00000000  s1: 00000000 ffffffff  t1: 003fe000  psw:  000000c0
a2: 00000005  s2: 00080a24 00000053  t2: 00000ff8  ipc:       000
a3: 00000003  s3: 00000000 00000021  t3: 00000000  ccr:    000000
a4: 00000004  s4: 00000000 000000fc  t4: 0000001b  cir:     0  tid: 00
a5: 00000002  s5: 00000000 00000001  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000001  s6: 00000000 8000c86e  t6: 0ad6bad7 vm_u:  00000000 00000000
a7: 00000007  s7: 00000000 00000000  t7: 0ad4bad5 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0
(spu)> sysreset
sysreset:   revision 5.1 (Mon Apr 27 11:43:53 1992)
(spu)> cpureg

Register dump for cpu: 0
a0: 00001000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 00012140  psw:  000000c0
a2: 00000005  s2: 00000000 00000001  t2: 00000000  ipc:       000
a3: 00000003  s3: ffffffff ffffffff  t3: 00000002  ccr:    000000
a4: 00000004  s4: 00000000 00000000  t4: 00000019  cir:     0  tid: 00
a5: 00000002  s5: 00000000 00000000  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000001  s6: a9a082a8 00000000  t6: 0ad6bad7 vm_u:  00000000 00000000
a7: 00000007  s7: 00000000 00000006  t7: 0ad4bad5 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

Register dump for cpu: 1
```

```
a0: 0b2a8eb4  s0: 00000000 00000000  t0: 003ffff8   pc:  00000000
a1: 00000000  s1: 00000000 ffffffff  t1: 003fe000  psw:  000000c0
a2: 00000005  s2: 00080a24 00000053  t2: 00000ff8  ipc:      000
a3: 00000003  s3: 00000000 00000021  t3: 00000000  ccr:    000000
a4: 00000004  s4: 00000000 000000fc  t4: 0000001b  cir:   0  tid: 00
a5: 00000002  s5: 00000000 00000001  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000001  s6: 00000000 8000c86e  t6: 0ad6bad7 vm_u:  00000000 00000000
a7: 00000007  s7: 00000000 00000000  t7: 0ad4bad5 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0
(spu)> cpureg -i 0
```

zapisanie do rejestrów procesora zaw except PSW ma jakąś wartość

```
Register dump for cpu: 0
a0: 00000000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 00000000  psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000000  ipc:      000
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:    000000
a4: 00000000  s4: 00000000 00000000  t4: 00000000  cir:   0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 00000000   vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 00000000 vm_u:  00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 00000000 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

Register dump for cpu: 1
a0: 00000000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 00000000  psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000000  ipc:      000
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:    000000
a4: 00000000  s4: 00000000 00000000  t4: 00000000  cir:   0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 00000000   vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 00000000 vm_u:  00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 00000000 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0
(spu)> cpureg -i ffffffff
```

zapisanie jedynek dajsc ff.. wcale nie zapisujemy samych jedynek

```
Register dump for cpu: 0
a0: a0a0a0a0  s0: b0b0b0b0 c0c0c0c0  t0: d0d0d0d0   pc:  00000000
a1: a1a1a1a1  s1: b1b1b1b1 c1c1c1c1  t1: d1d1d1d1  psw:  000000c0
a2: a2a2a2a2  s2: b2b2b2b2 c2c2c2c2  t2: d2d2d2d2  ipc:      000
a3: a3a3a3a3  s3: b3b3b3b3 c3c3c3c3  t3: d3d3d3d3  ccr:    000000
a4: a4a4a4a4  s4: b4b4b4b4 c4c4c4c4  t4: d4d4d4d4  cir:   0  tid: 00
a5: a5a5a5a5  s5: b5b5b5b5 c5c5c5c5  t5: d5d5d5d5   vl:  00   vs: 00000000
a6: a6a6a6a6  s6: b6b6b6b6 c6c6c6c6  t6: d6d6d6d6 vm_u:  00000000 00000000
a7: a7a7a7a7  s7: b7b7b7b7 c7c7c7c7  t7: d7d7d7d7 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

Register dump for cpu: 1
a0: a0a0a0a0  s0: b0b0b0b0 c0c0c0c0  t0: d0d0d0d0   pc:  00000000
a1: a1a1a1a1  s1: b1b1b1b1 c1c1c1c1  t1: d1d1d1d1  psw:  000000c0
a2: a2a2a2a2  s2: b2b2b2b2 c2c2c2c2  t2: d2d2d2d2  ipc:      000
a3: a3a3a3a3  s3: b3b3b3b3 c3c3c3c3  t3: d3d3d3d3  ccr:    000000
a4: a4a4a4a4  s4: b4b4b4b4 c4c4c4c4  t4: d4d4d4d4  cir:   0  tid: 00
a5: a5a5a5a5  s5: b5b5b5b5 c5c5c5c5  t5: d5d5d5d5   vl:  00   vs: 00000000
a6: a6a6a6a6  s6: b6b6b6b6 c6c6c6c6  t6: d6d6d6d6 vm_u:  00000000 00000000
a7: a7a7a7a7  s7: b7b7b7b7 c7c7c7c7  t7: d7d7d7d7 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0
(spu)> cpureg -i a5a5a5a5
```

duzo lepsze jest to podanie i później odwrotnie i przekonamy się czy nie ma przekłamania

bo 5 1010 2 0101

```
Register dump for cpu: 0
a0: a5a5a5a5  s0: a5a5a5a5 a5a5a5a5  t0: a5a5a5a5   pc:  00000000
a1: a5a5a5a5  s1: a5a5a5a5 a5a5a5a5  t1: a5a5a5a5  psw:  000000c0
a2: a5a5a5a5  s2: a5a5a5a5 a5a5a5a5  t2: a5a5a5a5  ipc:      000
a3: a5a5a5a5  s3: a5a5a5a5 a5a5a5a5  t3: a5a5a5a5  ccr:    000000
a4: a5a5a5a5  s4: a5a5a5a5 a5a5a5a5  t4: a5a5a5a5  cir:   0  tid: 00
a5: a5a5a5a5  s5: a5a5a5a5 a5a5a5a5  t5: a5a5a5a5   vl:  00   vs: 00000000
a6: a5a5a5a5  s6: a5a5a5a5 a5a5a5a5  t6: a5a5a5a5 vm_u:  00000000 00000000
a7: a5a5a5a5  s7: a5a5a5a5 a5a5a5a5  t7: a5a5a5a5 vm_l:  00000000 00000000
```

```
global_int_enab: ØØ  local_int_enab: ++  int_mode: ØØ  target_CPU: Ø  IUN: Ø

Register dump for cpu: 1
aØ: a5a5a5a5  sØ: a5a5a5a5 a5a5a5a5  tØ: a5a5a5a5   pc:  ØØØØØØØØ
a1: a5a5a5a5  s1: a5a5a5a5 a5a5a5a5  t1: a5a5a5a5  psw:  ØØØØØØØc Ø
a2: a5a5a5a5  s2: a5a5a5a5 a5a5a5a5  t2: a5a5a5a5  ipc:       ØØØ
a3: a5a5a5a5  s3: a5a5a5a5 a5a5a5a5  t3: a5a5a5a5  ccr:    ØØØØØØ
a4: a5a5a5a5  s4: a5a5a5a5 a5a5a5a5  t4: a5a5a5a5  cir:   Ø  tid: ØØ
a5: a5a5a5a5  s5: a5a5a5a5 a5a5a5a5  t5: a5a5a5a5   vl:  ØØ   vs: ØØØØØØØØ
a6: a5a5a5a5  s6: a5a5a5a5 a5a5a5a5  t6: a5a5a5a5 vm_u:  ØØØØØØØØ ØØØØØØØØ
a7: a5a5a5a5  s7: a5a5a5a5 a5a5a5a5  t7: a5a5a5a5 vm_l:  ØØØØØØØØ ØØØØØØØØ
global_int_enab: ØØ  local_int_enab: ff  int_mode: ØØ  target_CPU: Ø  ION: Ø
(spu)> cpureg -i 5a5a5a5a

Register dump for cpu: Ø
aØ: 5a5a5a5a  sØ: 5a5a5a5a 5a5a5a5a  tØ: 5a5a5a5a   pc:  ØØØØØØØØ
a1: 5a5a5a5a  s1: 5a5a5a5a 5a5a5a5a  t1: 5a5a5a5a  psw:  ØØØØØØØc Ø
a2: 5a5a5a5a  s2: 5a5a5a5a 5a5a5a5a  t2: 5a5a5a5a  ipc:       ØØØ
a3: 5a5a5a5a  s3: 5a5a5a5a 5a5a5a5a  t3: 5a5a5a5a  ccr:    ØØØØØØ
a4: 5a5a5a5a  s4: 5a5a5a5a 5a5a5a5a  t4: 5a5a5a5a  cir:   Ø  tid: ØØ
a5: 5a5a5a5a  s5: 5a5a5a5a 5a5a5a5a  t5: 5a5a5a5a   vl:  ØØ   vs: ØØØØØØØØ
a6: 5a5a5a5a  s6: 5a5a5a5a 5a5a5a5a  t6: 5a5a5a5a vm_u:  ØØØØØØØØ ØØØØØØØØ
a7: 5a5a5a5a  s7: 5a5a5a5a 5a5a5a5a  t7: 5a5a5a5a vm_l:  ØØØØØØØØ ØØØØØØØØ
global_int_enab: ØØ  local_int_enab: ff  int_mode: ØØ  target_CPU: Ø  ION: Ø
```

↑ widać w porównaniu z poprzednim, że zapis prawidłowy do rejestrów zmienia się kolejność a5 -> 5a OK.

```
Register dump for cpu: 1
aØ: 5a5a5a5a  sØ: 5a5a5a5a 5a5a5a5a  tØ: 5a5a5a5a   pc:  ØØØØØØØØ
a1: 5a5a5a5a  s1: 5a5a5a5a 5a5a5a5a  t1: 5a5a5a5a  psw:  ØØØØØØc Ø
a2: 5a5a5a5a  s2: 5a5a5a5a 5a5a5a5a  t2: 5a5a5a5a  ipc:       ØØØ
a3: 5a5a5a5a  s3: 5a5a5a5a 5a5a5a5a  t3: 5a5a5a5a  ccr:    ØØØØØØ
a4: 5a5a5a5a  s4: 5a5a5a5a 5a5a5a5a  t4: 5a5a5a5a  cir:   Ø  tid: ØØ
a5: 5a5a5a5a  s5: 5a5a5a5a 5a5a5a5a  t5: 5a5a5a5a   vl:  ØØ   vs: ØØØØØØØØ
a6: 5a5a5a5a  s6: 5a5a5a5a 5a5a5a5a  t6: 5a5a5a5a vm_u:  ØØØØØØØØ ØØØØØØØØ
a7: 5a5a5a5a  s7: 5a5a5a5a 5a5a5a5a  t7: 5a5a5a5a vm_l:  ØØØØØØØØ ØØØØØØØØ
global_int_enab: ØØ  local_int_enab: ff  int_mode: ØØ  target_CPU: Ø  ION: Ø
(spu)> cpureg -i Ø
```

zerowanie rej. cpu za wyjątkiem psw all łącznie z pc

```
Register dump for cpu: Ø
aØ: ØØØØØØØØ  sØ: ØØØØØØØØ ØØØØØØØØ  tØ: ØØØØØØØØ   pc:  ØØØØØØØØ
a1: ØØØØØØØØ  s1: ØØØØØØØØ ØØØØØØØØ  t1: ØØØØØØØØ  psw:  ØØØØØØØc Ø
a2: ØØØØØØØØ  s2: ØØØØØØØØ ØØØØØØØØ  t2: ØØØØØØØØ  ipc:       ØØØ
a3: ØØØØØØØØ  s3: ØØØØØØØØ ØØØØØØØØ  t3: ØØØØØØØØ  ccr:    ØØØØØØ
a4: ØØØØØØØØ  s4: ØØØØØØØØ ØØØØØØØØ  t4: ØØØØØØØØ  cir:   Ø  tid: ØØ
a5: ØØØØØØØØ  s5: ØØØØØØØØ ØØØØØØØØ  t5: ØØØØØØØØ   vl:  ØØ   vs: ØØØØØØØØ
a6: ØØØØØØØØ  s6: ØØØØØØØØ ØØØØØØØØ  t6: ØØØØØØØØ vm_u:  ØØØØØØØØ ØØØØØØØØ
a7: ØØØØØØØØ  s7: ØØØØØØØØ ØØØØØØØØ  t7: ØØØØØØØØ vm_l:  ØØØØØØØØ ØØØØØØØØ
global_int_enab: ØØ  local_int_enab: ff  int_mode: ØØ  target_CPU: Ø  ION: Ø
```

→ tylko to zostaje!

```
Register dump for cpu: 1
aØ: ØØØØØØØØ  sØ: ØØØØØØØØ ØØØØØØØØ  tØ: ØØØØØØØØ   pc:  ØØØØØØØØ
a1: ØØØØØØØØ  s1: ØØØØØØØØ ØØØØØØØØ  t1: ØØØØØØØØ  psw:  ØØØØØØc Ø
a2: ØØØØØØØØ  s2: ØØØØØØØØ ØØØØØØØØ  t2: ØØØØØØØØ  ipc:       ØØØ
a3: ØØØØØØØØ  s3: ØØØØØØØØ ØØØØØØØØ  t3: ØØØØØØØØ  ccr:    ØØØØØØ
a4: ØØØØØØØØ  s4: ØØØØØØØØ ØØØØØØØØ  t4: ØØØØØØØØ  cir:   Ø  tid: ØØ
a5: ØØØØØØØØ  s5: ØØØØØØØØ ØØØØØØØØ  t5: ØØØØØØØØ   vl:  ØØ   vs: ØØØØØØØØ
a6: ØØØØØØØØ  s6: ØØØØØØØØ ØØØØØØØØ  t6: ØØØØØØØØ vm_u:  ØØØØØØØØ ØØØØØØØØ
a7: ØØØØØØØØ  s7: ØØØØØØØØ ØØØØØØØØ  t7: ØØØØØØØØ vm_l:  ØØØØØØØØ ØØØØØØØØ
global_int_enab: ØØ  local_int_enab: ff  int_mode: ØØ  target_CPU: Ø  ION: Ø
(spu)> iscn sys
iscn:    revision 5.1 (Mon Apr 27 11:43:51 1992)

Initializing the symbol table for 1ØØØ entries
Initializing pcode space to 4ØØØ locations
Initializing p-machine stack to 1ØØØ locations
Including sys
```

```
Verify flag is now on
Including: /hw/cputest/halt_off

 Log_char returning EOF
Including: /hw/cputest/asp_func

 Log_char returning EOF
Including: /hw/cputest/load

 Log_char returning EOF
Including: /hw/cputest/clock

 Log_char returning EOF
Including: /hw/cputest/hard



Display hard error state of cpu with hard (head #)
Default Head [:900]:

 Log_char returning EOF
Including: /hw/cputest/tipr

 Log_char returning EOF
Default Head 1

Before Executing load (addr), TYPE !sysreset

 Log_char returning EOF
iscn68==>:900=1   PROCESOR NR 1

iscn68==>!sysreset
sysreset:   revision 5.1 (Mon Apr 27 11:43:53 1992)

 Hit <CR> to continue

iscn68==>!cpureg -c 1

Register dump for cpu: 1
a0: 00000000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 00000000  psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000000  ipc:      000
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:   000000
a4: 00000000  s4: 00000000 00000000  t4: 00000000  cir:    0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 00000000   vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 00000000 vm_u:  00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 00000000 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

 Hit <CR> to continue

iscn68==>load 120d0   załadowanie do pc adresu (force coldstart to start)
pc = 120d0    adres załadował się odpow. syst.
iscn68==>!cpureg -c 1

                                                    boot block
Register dump for cpu: 1
a0: 00000000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 000120d0 ok psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000000  ipc:      000
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:   680040
a4: 00000000  s4: 00000000 00000000  t4: 00000000  cir:    0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 00000000 ok vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 00000000 vm_u:  00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 00000000 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0
```

```
 Hit <CR> to continue

iscn68==>t
pc=00000000    ipc=040    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=04e    ep=000    ui=f    head 1
iscn68==>!cpureg -c 1

Register dump for cpu: 1
a0: 00000000  s0: 00000000 00000000  t0: 00000000   pc:  00000000
a1: 00000000  s1: 00000000 00000000  t1: 000120d0  psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000000  ipc:       04e
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:    680040
a4: 00000000  s4: 00000000 00000000  t4: 00000000  cir:    0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 00000000 vm_u:  00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 00000000 vm_l:  00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

 Hit <CR> to continue

iscn68==>tipr
 80
iscn68==>t
pc=00000000    ipc=080    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=080    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=080    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=080    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=080    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=081    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=068    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=069    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=06a    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=06b    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=06c    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=06d    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=06e    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=070    ep=000    ui=f    head 1
iscn68==>t
pc=00000000    ipc=071    ep=000    ui=f    head 1
iscn68==>t
pc=000120d0    ipc=c00    ep=000    ui=0    head 1
iscn68==>!cpureg -c 1

Register dump for cpu: 1
 0: 00000000  s0: 00000000 00000000  t0: 003ffff8   pc:  000120d0
a1: 00000000  s1: 00000000 00000000  t1: 003fe000  psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000ff8  ipc:       c00
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:    600000
a4: 00000000  s4: 00000000 00000000  t4: 0000001b  cir:    0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 0ad6bad7 vm_u:  00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 0ad4bad5 vm_l:  00000000 00000000
```

to be close to end of coldstart

przyśpieszenie ← nie dalekco do końca

2 reqTS na AS przez dfw g Amy adres rotadowy do PC na IP.

pc = t5 OK // Prawidłowo

nie na zer

OK

8

```
global_int_enab: ØØ  local_int_enab: ff  int_mode: ØØ  target_CPU: Ø  ION: Ø

 Hit <CR> to continue

iscn68==>t
pc=ØØØ12Ød Ø   ipc=cØØ   ep=ØØØ   ui=Ø   head 1
iscn68==>t
pc=ØØØ12Ød Ø   ipc=cØØ   ep=ØØØ   ui=Ø   head 1
iscn68==>in "hang"

Including: hang   took at glA1 dla proc. 0


Display hang state of the cpu with hang (head #)

Default Head [:9ØØ]:

 Log_char returning EOF

iscn68==>hang 1   dla proc. 1
          hang                    Cpu 1

IPP:      (pc)       (ba)       (na)          ja as_pend vp_pend inst_rdy
     ØØØ12Ød Ø ØØØØØØØØ ØØØ12Ød8 ØØØ12Ød Ø        Ø        Ø         Ø
           la la_maxed pur_state
     ØØØ12eØ        Ø         5
next address dla proc 1.
ASP: ipc upc ui_valid ui_lvl_1 ip_inh mem_idle fu_busy vc_idle
     cØØ cØ1        Ø        Ø      Ø        Ø       Ø        1

SFU: vl       vs vg_busy1 vg_dbusy1 vg_vlc vg_dvlc vg_qmop vg_qsiz
     ØØ ØØØØØØØØ        Ø         Ø      ff       Ø       Ø       Ø
     mmqempty     vl_rdy_as     vl_rdy_vc  vmq_ptr  vpq_ptr
            1             Ø             Ø        Ø        Ø

DCU: qreq qfault qlab_full qrd_cyc qwr_cyca qex_cyc      qla1
        1      1          1       1        Ø       1      ØØØ12Ød Ø  to samo OK

VPC: (disp) ip_disp (ep) ul_active_1 ul_uia ua_active ua_uia um_active um_uia
          Ø       Ø  ØØØ           Ø    ØØØ         Ø    ØØØ         Ø    ØØØ
     lbd_state dlp_state os_state_reg vmo_state_reg
             Ø         Ø            Ø            Ø

iscn68==>t
pc=ØØØ12Ød Ø   ipc=Ø11   ep=ØØØ   ui=f   head 1   to oznacza, że program tutaj idzie do Programu PTE MISS
iscn68==>t
pc=ØØØ12Ød Ø   ipc=11Ø   ep=ØØØ   ui=f   head 1
iscn68==> t 15
pc=ØØØ12Ød Ø   ipc=114   ep=ØØØ   ui=f   head 1
iscn68==>in "fu_mmqueue"

Including: fu_mmqueue


Message: Dump mmqueue with dump_mmq (head #)

 .og_char returning EOF

iscn68==>dump_mmq 1

MMQUEUE     Head 1
NUM                        POP       part qinh  MOx_EN  MEx_EN  CU
ITEMS _qpush empty full    q qq qqq     2  sxv  3 2 1 Ø 3 2 1 Ø EN
_____ ______ _____ ____    _________  ____ ____  _______ _______ __
```

9

```
       Ø      1      1     Ø   Ø   1    Ø      Ø     Ø   Ø Ø Ø 1 Ø Ø Ø 1    Ø

 o  loc  or  bd  qaØ  qa1        req                          full  op       mmq  cu  mxv   inh
 k  ptr   #  Ø1  <<3  <<3        Ø1*  sz  rot  unit   rtn      upd  rdy  vm  act  en  dual  ust
 -  ---  --  --  ---  --------   ---  --  ---  ----   ---      ---  ---  --  ---  --  ----  ---

   wp-> 2 (next position to be written)
 N qw-> 2 11 ØdØ ØØØ12ØdØ 11    3    Ø 2 IP   Øf  T7      1    1   1    Ø    Ø     Ø    Ø
 N rp-> 2 ØØ f9Ø ØØ3fff9Ø 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø

 N qr-> 2 ØØ f9Ø ØØ3fff9Ø 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      1 11 ØdØ Øad6bØdØ Ø1    2    4 3 ASn  Øf  T7      1    1   1    Ø    Ø     Ø    Ø
 N      Ø ØØ ØdØ Øad6bØdØ Ø1    2    4 3 ASn  Øf  T7      1    1   1    Ø    Ø     Ø    Ø
 N      f 11 ff8 ØØ3ffff8 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      e 11 ffØ ØØ3ffffØ 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      d 11 fe8 ØØ3fffe8 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      c 11 feØ ØØ3fffeØ 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      b 11 fd8 ØØ3fffd8 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      a 11 fdØ ØØ3fffdØ 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø
 N      9 11 fc8 ØØ3fffc8 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      8 11 fcØ ØØ3fffcØ 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      7 ØØ fb8 ØØ3fffb8 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      6 ØØ fbØ ØØ3fffbØ 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      5 ØØ fa8 ØØ3fffa8 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      4 ØØ faØ ØØ3fffaØ 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
 N      3 ØØ f98 ØØ3fff98 11    3    Ø 2 IP   Ø8  TØ      1    1   1    Ø    Ø     Ø    Ø
  \ NOT VALID

 iscn68==>in "asp_func"

 Including: asp_func

  Log_char returning EOF

 iscn68==>ga                                   => można sprawdzić rozpisując czy
                                                  maszyna dobrze robi parity
 ASP              DFW Gate Array Dump   par. ok    Head 1
                                        odd
 ---------------------------------------001 = odd------------------------------------
 DFW:    ureg_u       ureg_l     ureg_par     dcreg_u     dcreg_l     dcreg_par
        ffffffff     ffffffff      ff        ØØØØØØØØ    ØØØØØØØØ        ff
 słowo  ->
 danych  sregØ_u      sregØ_l     sregØ_par    sreg1_u     sreg1_l     sreg1_par
 w nery ØØØØØØØØ     ØØØØØØØØ      ff        ØØØØØØØØ    ØØØØØØØØ        ff

  u.[8]   u.[7]   u.[6]   u.[5]   u.[4]   u.[3]   u.[2]   u.[1]   u.[Ø]
   ff      ff      ff      ff      ff      ff      ff      ff      ff
  dc.[8]  dc.[7]  dc.[6]  dc.[5]  dc.[4]  dc.[3]  dc.[2]  dc.[1]  dc.[Ø]
   ff      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ
  s1.[8]  s1.[7]  s1.[6]  s1.[5]  s1.[4]  s1.[3]  s1.[2]  s1.[1]  s1.[Ø]
   ff      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ
  sØ.[8]  sØ.[7]  sØ.[6]  sØ.[5]  sØ.[4]  sØ.[3]  sØ.[2]  sØ.[1]  sØ.[Ø]
   ff      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ      ØØ

 iscn68==>t 15
 pc=ØØØ121Ø4   ipc=c23   ep=Ø93   ui=f   head 1
 iscn68==>t
 pc=ØØØ12ØdØ   ipc=118   ep=ØØØ   ui=f   head 1
 iscn68==>t
 pc=ØØØ12ØdØ   ipc=Ø11   ep=ØØØ   ui=Ø   head 1
 iscn68==>ga                          \ NOT VALID => ok Bo Adress wówczas ogólnie już fizyczny
                                               z wyjścia PTE
 ASP              DFW Gate Array Dump              Head 1
 ------------------------------------------------------------------------------------
 DFW:    ureg_u       ureg_l     ureg_par     dcreg_u     dcreg_l     dcreg_par
        ffffffff     ffffffff      ff        ØØØØØØØØ    ØØØØØØØØ        ff
         sregØ_u      sregØ_l     sregØ_par    sreg1_u     sreg1_l     sreg1_par
        ØØØØØØØØ     ØØØØØØØØ      ff        ØØØØØØØØ    ØØØØØØØØ        ff

  u.[8]   u.[7]   u.[6]   u.[5]   u.[4]   u.[3]   u.[2]   u.[1]   u.[Ø]
   ff      ff      ff      ff      ff      ff      ff      ff      ff
  dc.[8]  dc.[7]  dc.[6]  dc.[5]  dc.[4]  dc.[3]  dc.[2]  dc.[1]  dc.[Ø]
```

```
   ff       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ
 s1.[8]   s1.[7]   s1.[6]   s1.[5]   s1.[4]   s1.[3]   s1.[2]   s1.[1]   s1.[Ø]
   ff       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ
 sØ.[8]   sØ.[7]   sØ.[6]   sØ.[5]   sØ.[4]   sØ.[3]   sØ.[2]   sØ.[1]   sØ.[Ø]
   ff       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ       ØØ

iscn68==>!mm
mm:   revision 5.1 (Mon Apr 27 11:43:52 1992)
mm:address mode = PHYSICAL
mm(ØØ,ØØ):s log
                        Current sdrs:

sdr[Ø]=23fffe1Ø sdr[1]=23fffc1Ø sdr[2]=23fffa1Ø sdr[3]=23fff81Ø
sdr[4]=23fff61Ø sdr[5]=23fff41Ø sdr[6]=23fff21Ø sdr[7]=23fff01Ø
mm(ØØ,ØØ):d 120dØ,4Ø
ØØØ120dØ 11 c1 55 55 55 55 11 c2 aa aa aa aa 11 83 ff ff    ..UUUU.. ........
ØØØ120eØ 11 c4 ee ee ee ee 11 c5 77 77 77 77 11 c6 99 99    ........ wwww....
ØØØ120fØ 99 99 11 c7 66 66 66 66 11 c1 aa aa aa aa 11 c2    ....ffff ........
ØØØ121ØØ 55 55 55 55 44 c3 11 c4 11 11 11 11 11 c5 88 88    UUUUD... ........
mm(ØØ,ØØ):q

 Hit <CR> to continue

iscn68==>in "mcm_func"

Including: mcm_func



Type howto_mcm_func for information using these scripts.


Message: Dump win queue with dump_win_q(mcm)
           where mcm is the slot number to be dumped.

Message: Dump read queue with dump_rd_q(port,mcm)
           where port is the port to be dumped
           and mcm is the slot number to be dumped.

Message:Display xbar rd/wr pointers with xbar_ptr (port,mcm)

Message: Decode mcm errors with mcm_err().

 Log_char returning EOF

iscn68==>dump_rd_q 1,Ø
-- XBAR read queue contents for port B of MCM[Ø] --
rd_ptr    wr_ptr    rd_par     rd_dat
   2         3         f      ffffffff     (current)
   2         3         f      ffffffff     (after 1 clock)
   1         3         4      23ffe01f
  8Ø         3         f      ffffffff
  4Ø         3         f      ffffffff
  2Ø         3         f      ffffffff
  1Ø         3         f      ffffffff
   8         3         f      ffffffff
   4         3         f      ffffffff

  rcnt = Ø       dcnt = Ø

iscn68==>dump_rd_q 1,1
-- XBAR read queue contents for port B of MCM[1] --
rd_ptr    wr_ptr    rd_par     rd_dat
   1         1         f      ffffffff     (current)
   1         1         f      ffffffff     (after 1 clock)
  8Ø         1         f      ffffffff
```

```
  4Ø          1          +      ++++++++
  2Ø          1          f      ffffffff
  1Ø          1          f      ffffffff
   8          1          f      ffffffff
   4          1          f      ffffffff
   2          1          f      ffffffff

  rcnt = Ø       dcnt = Ø

iscn68==>t
pc=ØØØ12Ød0   ipc=cØØ   ep=ØØØ   ui=Ø   head 1
iscn68==>ga
ASP                DFW Gate Array Dump          Head 1
------------------------------------------------------------------------------
DFW:   ureg_u       ureg_l   ureg_par   dcreg_u   dcreg_l    dcreg_par
       ffffffff ≠ ffffffff      ff      ØØØØØØØØ   ØØØØØØØØ      ff
   sregØ_u      sregØ_l   sregØ_par   sreg1_u    sreg1_l    sreg1_par
   ØØØØØØØØ     ØØØØØØØØ     ff       ØØØØØØØØ   ØØØØØØØØ      ff

 u.[8]  u.[7]  u.[6]  u.[5]  u.[4]  u.[3]  u.[2]  u.[1]  u.[Ø]
   ff     ff     ff     ff     ff     ff     ff     ff     ff
 dc.[8] dc.[7] dc.[6] dc.[5] dc.[4] dc.[3] dc.[2] dc.[1] dc.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 s1.[8] s1.[7] s1.[6] s1.[5] s1.[4] s1.[3] s1.[2] s1.[1] s1.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 sØ.[8] sØ.[7] sØ.[6] sØ.[5] sØ.[4] sØ.[3] sØ.[2] sØ.[1] sØ.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ

iscn68==>dump_rd_q 1,Ø
-- XBAR read queue contents for port B of MCM[Ø] --
rd_ptr    wr_ptr    rd_par    rd_dat
   2         3         f      ffffffff    (current)
   2         3         f      ffffffff    (after 1 clock)
   1         3         4      23ffeØ1f
  8Ø         3         f      ffffffff
  4Ø         3         f      ffffffff
  2Ø         3         f      ffffffff
  1Ø         3         f      ffffffff
   8         3         f      ffffffff
   4         3         f      ffffffff

  rcnt = Ø       dcnt = Ø

iscn68==>dump_rd_q 1,1
-- XBAR read queue contents for port B of MCM[1] --
rd_ptr    wr_ptr    rd_par    rd_dat
   1         1         f      ffffffff    (current)
   1         1         f      ffffffff    (after 1 clock)
  8Ø         1         f      ffffffff
  4Ø         1         f      ffffffff
  2Ø         1         f      ffffffff
  1Ø         1         f      ffffffff
   8         1         f      ffffffff
   4         1         f      ffffffff
   2         1         f      ffffffff

  rcnt = Ø       dcnt = Ø

 scn68==>in "mcm_scr"

Including: mcm_scr

DESCRIPTION: Fixes problems that occur because the screens
for mcm's and mcm3's are different.  mcm_scr has been automated to
bring up the appropriate error screen depending on the type of memory
boards installed. A good choice for mixed memory systems.
```

```
Type howto_mcm_scr for information on invoking mcm, mcm3 screens.

Message: Defining port (slot)

Message: Defining all_ports (slot)

Message: Defining error (slot)

 Log_char returning EOF

iscn68==>port 1
m
v
s
m
v
s
m
v
s
m
v
s
m
v
s
m
v
s
m
v
m  rd_par   rd_dat rd_ptr wr_ptr
m  rdy ov_cycle ov_bank ov_row ov_adr ov_zone ov_wr_par ov_wr_dat
v
s
m
v
s
m  req    cycle    bank    row    adr    zone    wr_par    wr_dat
m  board                          port                          check
v      1                             4                            Ø
v      Ø                             1                            Ø
v      1                             1                            Ø
v      Ø                             1                            Ø

iscn68==>port Ø
m
v
s
m
v
s
m
v
s
m
v
s
m
v
s
m
v
```

```
s
m
v
m  rd_par    rd_dat rd_ptr wr_ptr
m  rdy ov_cycle ov_bank ov_row ov_adr ov_zone ov_wr_par ov_wr_dat
v
s
m
v
s
m  req     cycle     bank     row      adr      zone      wr_par      wr_dat
m  board                              port                                  check
v      Ø                                 1                                     Ø

iscn68==>t
pc=ØØØ12Ød8    ipc=cØØ    ep=ØØØ    ui=Ø    head 1
iscn68==>ga
ASP                 DFW Gate Array Dump          Head 1
-------------------------------------------------------------------------------
DFW:    ureg_u        ureg_l    ureg_par    dcreg_u    dcreg_l     dcreg_par
        ffffffff idle ffffffff      ff      ØØØØØØØØ   ØØØØØØØØ       ff
    sregØ_u       sregØ_l    sregØ_par    sreg1_u    sreg1_l     sreg1_par
    ØØØØØØØØ      ØØØØØØØØ      ff       ØØØØØØØØ   ØØØØØØØØ       ff

 u.[8]  u.[7]  u.[6]  u.[5]  u.[4]  u.[3]  u.[2]  u.[1]  u.[Ø]
   ff     ff     ff     ff     ff     ff     ff     ff     ff
 dc.[8] dc.[7] dc.[6] dc.[5] dc.[4] dc.[3] dc.[2] dc.[1] dc.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 s1.[8] s1.[7] s1.[6] s1.[5] s1.[4] s1.[3] s1.[2] s1.[1] s1.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 sØ.[8] sØ.[7] sØ.[6] sØ.[5] sØ.[4] sØ.[3] sØ.[2] sØ.[1] sØ.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ

iscn68==>dump_rd_q 1,Ø
-- XBAR read queue contents for port B of MCM[Ø] --
rd_ptr    wr_ptr    rd_par    rd_dat
   2         3         f     ffffffff    (current)
   2         3         f     ffffffff    (after 1 clock)
   1         3         4     23ffe01f
  8Ø         3         f     ffffffff
  4Ø         3         f     ffffffff
  2Ø         3         f     ffffffff
  1Ø         3         f     ffffffff
   8         3         f     ffffffff
   4         3         f     ffffffff

  rcnt = Ø       dcnt = Ø

iscn68==>dump_rd_q 1,1
-- XBAR read queue contents for port B of MCM[1] --
rd_ptr    wr_ptr    rd_par    rd_dat
   1         1         f     ffffffff    (current)
   1         1         f     ffffffff    (after 1 clock)
  8Ø         1         f     ffffffff
  4Ø         1         f     ffffffff
  2Ø         1         f     ffffffff
  1Ø         1         f     ffffffff
   8         1         f     ffffffff
   4         1         f     ffffffff
   2         1         f     ffffffff

  rcnt = Ø       dcnt = Ø

iscn68==>t
pc=ØØ12Ød8    ipc=cØØ    ep=ØØØ    ui=Ø    head 1
iscn68==>ga
```

ok

```
iscn68==>ga
ASP                DFW Gate Array Dump          Head 1
-------------------------------------------------------------------------------
DFW:    ureg_u      ureg_l    ureg_par    dcreg_u    dcreg_l     dcreg_par
       ffffffff    ffffffff      ff      ØØØØØØØØ   ØØØØØØØØ       ff
   sregØ_u      sregØ_l    sregØ_par    sreg1_u    sreg1_l     sreg1_par
   ØØØØØØØØ     ØØØØØØØØ      ff       ØØØØØØØØ   ØØØØØØØØ       ff

 u.[8]  u.[7]  u.[6]  u.[5]  u.[4]  u.[3]  u.[2]  u.[1]  u.[Ø]
   ff     ff     ff     ff     ff     ff     ff     ff     ff
 dc.[8] dc.[7] dc.[6] dc.[5] dc.[4] dc.[3] dc.[2] dc.[1] dc.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 s1.[8] s1.[7] s1.[6] s1.[5] s1.[4] s1.[3] s1.[2] s1.[1] s1.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 sØ.[8] sØ.[7] sØ.[6] sØ.[5] sØ.[4] sØ.[3] sØ.[2] sØ.[1] sØ.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ

iscn68==>t 1Ø
pc=ØØØ12Ød8   ipc=cØØ   ep=ØØØ   ui=Ø   head 1
iscn68==>ga
ASP                DFW Gate Array Dump          Head 1
-------------------------------------------------------------------------------
DFW:    ureg_u      ureg_l    ureg_par    dcreg_u    dcreg_l     dcreg_par
       ffffffff    ffffffff      ff      ØØØØØØØØ   ØØØØØØØØ       ff
   sregØ_u      sregØ_l    sregØ_par    sreg1_u    sreg1_l     sreg1_par
   ØØØØØØØØ     ØØØØØØØØ      ff       ØØØØØØØØ   ØØØØØØØØ       ff

 u.[8]  u.[7]  u.[6]  u.[5]  u.[4]  u.[3]  u.[2]  u.[1]  u.[Ø]
   ff     ff     ff     ff     ff     ff     ff     ff     ff
 dc.[8] dc.[7] dc.[6] dc.[5] dc.[4] dc.[3] dc.[2] dc.[1] dc.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 s1.[8] s1.[7] s1.[6] s1.[5] s1.[4] s1.[3] s1.[2] s1.[1] s1.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 sØ.[8] sØ.[7] sØ.[6] sØ.[5] sØ.[4] sØ.[3] sØ.[2] sØ.[1] sØ.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ

iscn68==>ga
ASP                DFW Gate Array Dump          Head 1
-------------------------------------------------------------------------------
DFW:    ureg_u      ureg_l    ureg_par    dcreg_u    dcreg_l     dcreg_par
      11c15555 ok. 555511c2      be      ØØØØØØØØ   ØØØØØØØØ       ff
   sregØ_u      sregØ_l    sregØ_par    sreg1_u    sreg1_l     sreg1_par
   ØØØØØØØØ     ØØØØØØØØ      ff       ØØØØØØØØ   ØØØØØØØØ       ff

 u.[8]  u.[7]  u.[6]  u.[5]  u.[4]  u.[3]  u.[2]  u.[1]  u.[Ø]
   be     41     7d     ØØ     be     ØØ     3c     Ø1     fe
 dc.[8] dc.[7] dc.[6] dc.[5] dc.[4] dc.[3] dc.[2] dc.[1] dc.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 s1.[8] s1.[7] s1.[6] s1.[5] s1.[4] s1.[3] s1.[2] s1.[1] s1.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 sØ.[8] sØ.[7] sØ.[6] sØ.[5] sØ.[4] sØ.[3] sØ.[2] sØ.[1] sØ.[Ø]
   ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ

iscn68==>dump_rd_q 1,Ø
-- XBAR read queue contents for port B of MCM[Ø] --
rd_ptr    wr_ptr    rd_par    rd_dat
   2         3         f     ffffffff    (current)
   2         3         f     ffffffff    (after 1 clock)
   1         3         4     23ffeØ1f
  8Ø         3         f     ffffffff
  4Ø         3         f     ffffffff
  2Ø         3         f     ffffffff
  1Ø         3         f     ffffffff
   8         3         f     ffffffff
   4         3         f     ffffffff
```

```
  rcnt = Ø          dcnt = Ø

iscn68==>dump_rd_q 1,1
-- XBAR read queue contents for port B of MCM[1] --
rd_ptr     wr_ptr     rd_par     rd_dat
   1          1          f       ffffffff     (current)
   1          1          f       ffffffff     (after 1 clock)
  8Ø          1          f       ffffffff
  4Ø          1          f       ffffffff
  2Ø          1          f       ffffffff
  1Ø          1          f       ffffffff
   8          1          f       ffffffff
   4          1          f       ffffffff
   2          1          f       ffffffff

  rcnt = Ø          dcnt = Ø

iscn68==>dump_rd_q 1,2
-- XBAR read queue contents for port B of MCM[2] --
rd_ptr     wr_ptr     rd_par     rd_dat
   4          f          b       11c15555     (current)       Ok. data was move.
   8          e          f       aaaaaaaa     (after 1 clock)
   4          e          f       aaaaaaaa
   2          e          b       11c15555
   1          e          8         20801f
  8Ø          e          f       ffffffff
  4Ø          e          f       ffffffff
  2Ø          e          f       ffffffff
  1Ø          e          f       ffffffff

  rcnt = 2          dcnt = 1

iscn68==>ga
ASP                DFW Gate Array Dump          Head 1
------------------------------------------------------------------------
DFW:   ureg_u        ureg_l     ureg_par    dcreg_u     dcreg_l     dcreg_par
     11c15555 OK   555511c2 *     be       ØØØØØØØØ    ØØØØØØØØ        ff
   sregØ_u        sregØ_l     sregØ_par    sreg1_u     sreg1_l     sreg1_par
   ØØØØØØØØ       ØØØØØØØØ       ff        ØØØØØØØØ    ØØØØØØØØ        ff

 u.[8]  u.[7]  u.[6]  u.[5]  u.[4]  u.[3]  u.[2]  u.[1]  u.[Ø]
  be     41     7d     ØØ     be     ØØ     3c     Ø1     fe
 dc.[8] dc.[7] dc.[6] dc.[5] dc.[4] dc.[3] dc.[2] dc.[1] dc.[Ø]
  ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 s1.[8] s1.[7] s1.[6] s1.[5] s1.[4] s1.[3] s1.[2] s1.[1] s1.[Ø]
  ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ
 sØ.[8] sØ.[7] sØ.[6] sØ.[5] sØ.[4] sØ.[3] sØ.[2] sØ.[1] sØ.[Ø]
  ff     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ     ØØ

iscn68==>in "ipp_func"

Including: ipp_func
                                                   ↑ tylko tam cały wyrez

To print the mdat reg. use m_dat <head #>
                                                   ↓ a tutaj z dwóch ...ów
To read Icache use rd_ic <start addr>,<end addr>     ... że to to samo
                                                     o tym wie logik
 rite Icache wr_ic <wr addr>,<hwØ>,<hw1>,<hw2>,<hw3>

 IPP hang state use ipp_hang <head #>

 Log_char returning EOF          in data latch is the same

iscn68==>m_dat 1     ie w Data latch on IP 2 jego parity           15
m_datØ= 11c1   m_parØ= 1 Ø    m_dat1= 5555 * m_par1= 1 1
```

```
m_dat2= 5555  m_par2= 1 1   m_dat3= 11c2  m_par3= 1 0

iscn68==>rd_ic 120d0,120e0
            inst_0   inst_1   inst_2   inst_3          err  tag   stat a b
000120d0 = 0000 2 0 0000 2 0 0000 2 0 0000 2 0  MISS 0000 001ff 0005 0 0
                  -- Lookahead --                MISS 0    001ff      0 0
000120d8 = 0000 2 0 0000 2 0 0000 2 0 0000 2 0  MISS 0000 001ff 0005 0 0
                  -- Lookahead --                MISS 0    001ff      0 0
000120e0 = 0000 2 0 0000 2 0 0000 2 0 0000 2 0  MISS 0000 001ff 0005 0 0
                  -- Lookahead --                MISS 0    001ff      0 0

iscn68==>t
pc=000120d0   ipc=c00   ep=000   ui=0   head 1
iscn68==>rd_ic 120d0,120e0
            inst_0   inst_1   inst_2   inst_3          err  tag   stat a b
000120d0 = 11c1 3 0 5555 1 0 5555 1 0 11c2 3 0  HIT  0000 00009 0005 1 0
                  -- Lookahead --                HIT  0    00009      1 0
000120d8 = 0000 2 0 0000 2 0 0000 2 0 0000 2 0  MISS 0000 001ff 0005 0 0
                  -- Lookahead --                MISS 0    001ff      0 0
000120e0 = 0000 2 0 0000 2 0 0000 2 0 0000 2 0  MISS 0000 001ff 0005 0 0
                  -- Lookahead --                MISS 0    001ff      0 0

iscn68==>t
pc=000120d0   ipc=c00   ep=023   ui=0   head 1
iscn68==>t
pc=000120d6   ipc=c23   ep=023   ui=f   head 1
iscn68==>!cpureg -c 1

Register dump for cpu: 1
a0: 00000000  s0: 00000000 00000000  t0: 003ffff8   pc:  000120d6
a1: 00000000  s1: 00000000 00000000  t1: 003fe000  psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000ff8  ipc:       c23
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:    600000
a4: 00000000  s4: 00000000 00000000  t4: 0000001b  cir:    0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 0ad6bad7 vm_u: 00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 0ad4bad5 vm_l: 00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

 Hit <CR> to continue

iscn68==>t
pc=000120dc   ipc=c23   ep=023   ui=f   head 1
iscn68==>!cpureg -c 1

Register dump for cpu: 1
a0: 00000000  s0: 00000000 00000000  t0: 003ffff8   pc:  000120dc
a1: 55555555  s1: 00000000 00000000  t1: 003fe000  psw:  000000c0
a2: 00000000  s2: 00000000 00000000  t2: 00000ff8  ipc:       c23
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:    600000
a4: 00000000  s4: 00000000 00000000  t4: 0000001b  cir:    0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 000120d0   vl:  00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 0ad6bad7 vm_u: 00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 0ad4bad5 vm_l: 00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

 Hit <CR> to continue

 iscn68==>t
pc=000120e0   ipc=c23   ep=023   ui=f   head 1
iscn68==>!cpureg -c 1

Register dump for cpu: 1
a0: 00000000  s0: 00000000 00000000  t0: 003ffff8   pc:  000120e0
a1: 55555555  s1: 00000000 00000000  t1: 003fe000  psw:  000000c0
a2: aaaaaaaa  s2: 00000000 00000000  t2: 00000ff8  ipc:       c23
```

W Icache nothing

To samo co w Icache'u

zmienia się tutaj ep

przedostrzi na IPC

16

```
a3: 00000000  s3: 00000000 00000000  t3: 00000000  ccr:    600000
a4: 00000000  s4: 00000000 00000000  t4: 0000001b  cir:     0  tid: 00
a5: 00000000  s5: 00000000 00000000  t5: 00012000   vl:   00   vs: 00000000
a6: 00000000  s6: 00000000 00000000  t6: 0ad6bad7 vm_u:   00000000 00000000
a7: 00000000  s7: 00000000 00000000  t7: 0ad4bad5 vm_l:   00000000 00000000
global_int_enab: 00  local_int_enab: ff  int_mode: 00  target_CPU: 0  ION: 0

 Hit <CR> to continue

iscn68==>q
```