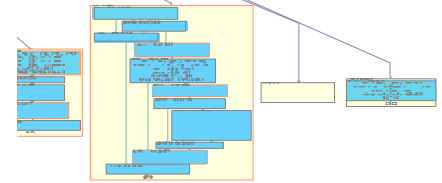
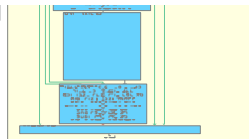
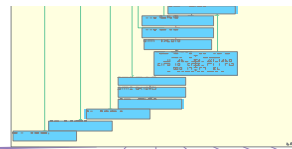
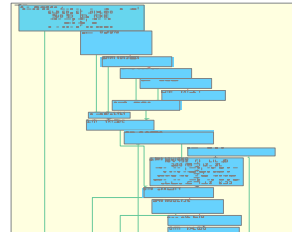


# Generic Slicing on Machine Code



**Marc Schlickling**  
[schlickling@cs.uni-sb.de](mailto:schlickling@cs.uni-sb.de)  
<http://rw4.cs.uni-sb.de/~schlickling>



# Embedded Systems (1)

---



- Embedded Systems (ES) are widely used
  - Many systems of daily use: handy, handheld, pda, ...
  - Safety critical systems: airbag control, flight control system, ...
- Rapidly growing complexity of software in ES
- But what happens, if such a system crashes?

# Embedded Systems (1)

Hello?



- Embedded Systems (ES) are widely used
  - Many systems of daily use: handy, handheld, pda, ...
  - Safety critical systems: airbag control, flight control system, ...
- Rapidly growing complexity of software in ES
- But what happens, if such a system crashes?

# Embedded Systems (1)

Hello?



- Embedded Systems (ES) are widely used
  - Many systems of daily use: handy, handheld, pda, ...
  - Safety critical systems: airbag control, flight control system, ...
- Rapidly growing complexity of software in ES
- But what happens, if such a system crashes?

# Embedded Systems (1)



- Embedded Systems (ES) are widely used
  - Many systems of daily use: handy, handheld, pda, ...
  - Safety critical systems: airbag control, flight control system, ...
- Rapidly growing complexity of software in ES
- But what happens, if such a system crashes?

# Embedded Systems (2)

---

- ES subject to strict safety constraints
  - Strict timing constraints
    - Execution has to be always fast enough
  - Strict resource constraints
    - No stack overflow during run time
  - Strict spacial partitioning
    - Different tasks run within their own memory spaces
- Urgent need to guarantee these characteristics

# Guaranteeing safety constraints (1)

---

- Not only possible at source code level
  - Caused by compilers
    - Optimizations
    - Transformations
- Complex due to the use of integrated modular platforms
  - E.g. IMA (integrated modular avionics)
    - Many components sharing the same resource
    - Used in the new Airbus A380

# Guaranteeing safety constraints (2)

---

- Programs checking such constraints
  - aiT
    - Worst case execution time (WCET) prediction
  - StackAnalyzer
    - Worst case stack usage determination
- Analyses are done on real executable
  - Due to the complexity of analyzed software, some help of the developers required



# Appliance of slicing

---

- Program understanding
  - Hard at machine code level
- Traceability
  - Is a branch in the control flow taken or not?
  - Ensure, that a specific variable in the program is not changed by other parts
    - E.g. co-driver airbag module

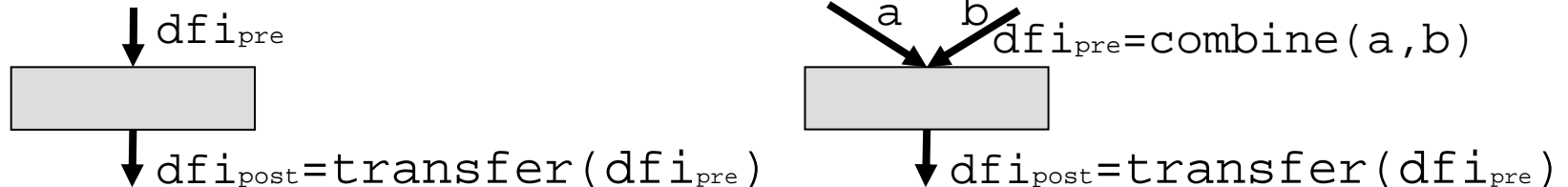
# In this talk

---

- Interprocedural backward slicing on machine code
- Based on data flow
  - Data dependencies
    - Only flow dependencies required
  - Control dependencies
- Setup
  - Textual representation of a control flow graph + some additional attributes
  - Results of a value analysis
    - Bounds of memory accesses

# Data flow analysis/Abstract interpretation

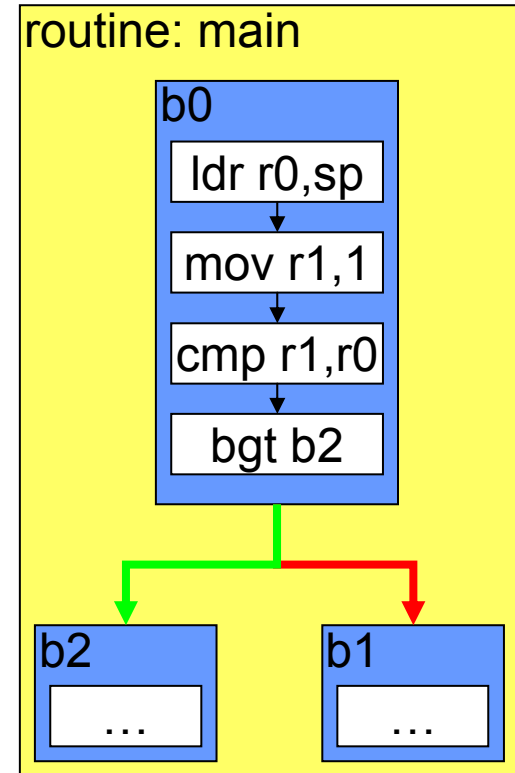
- A technique for collecting run-time information about data in programs without actually executing them
- Based on a control flow graph



- Abstract interpretation: computation on abstract values

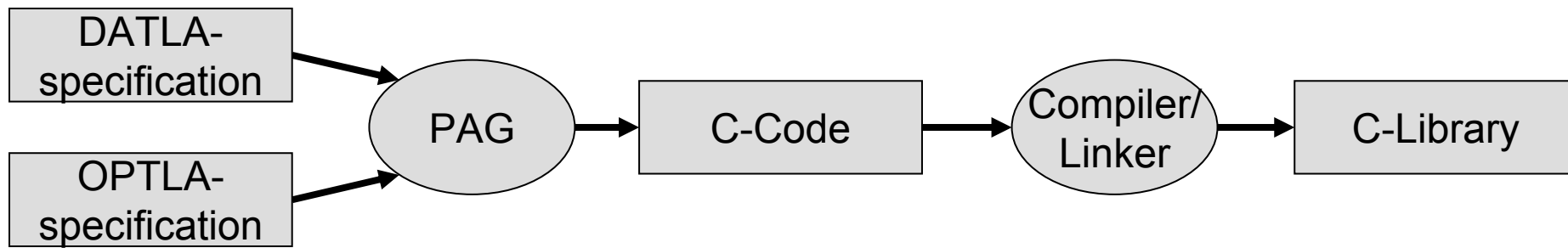
# Control Flow Graph & CRL

```
start with main;
routine main: name="main"
{ entry b0: id="b0"
  { edges to b1/f , b2/t;
  contains {
    0x00 "ldr r0,sp,0": src1="al",dst2="
src3="sp", src4="Mem", guard="always";
    0x04 "mov r1,1": src1="al",dst2="
src3="1", guard="always";
    0x08 "cmp r1,r0": src1="al",src2="
src3="r0",dst4="CPSR",
guard="always";
    0x0C "bgt b2": src1="gt",src2="0x14",
dst3="PC",guard="conditional";
  }
}
```



# Program Analyzer Generator (PAG)

- Analogous to flex and bison
- For generating efficient program analyzers
- CRL frontend available to translate CRL files into data structures describing the CFG
- Interface to read results from other analyses and using them in the current one



# Reconstruct data dependencies

---

- Simple reaching definition analysis
- Hardware-specific problems
  - Handling of composed registers / register aliasing
    - Architectures provide registers or register parts under different names
  - Guarded execution
    - Execution of an instruction depends on a condition
      - Instruction is definitively executed: must-update
      - Instruction may be executed: may-update

# Reconstruct control dependencies

---

- Computed by two consecutive analyses
  - Control point analysis (cp)
    - Forward directed
    - Collecting all nodes with more than one successor
  - Post dominator analysis (pdom)
    - Backward directed
- Combination of the results
  - For a node  $n$ , all nodes,  $n$  is dependent on, get computed by:  $\{m \mid m \in \text{dfi}_{cp}(n) : n \notin \text{dfi}_{pdom}(m)\}$

# Model memory accesses

- Consider the following example with slicing criterion  $C = (5, R1)$

```
M[0] = 1;  
M[1] = 2;  
R1 = M[0];  
R2 = M[1];  
R2 = R1 + R2;  
M[2] = R2;
```

input program

```
M[1] = 2;  
R1 = M[0];  
  
R2 = R1 + R2;
```

wrong slice

```
M[0] = 1;  
M[1] = 2;  
R1 = M[0];  
  
R2 = R1 + R2;
```

conservative slice

```
M[0] = 1;  
  
R1 = M[0];  
  
R2 = R1 + R2;
```

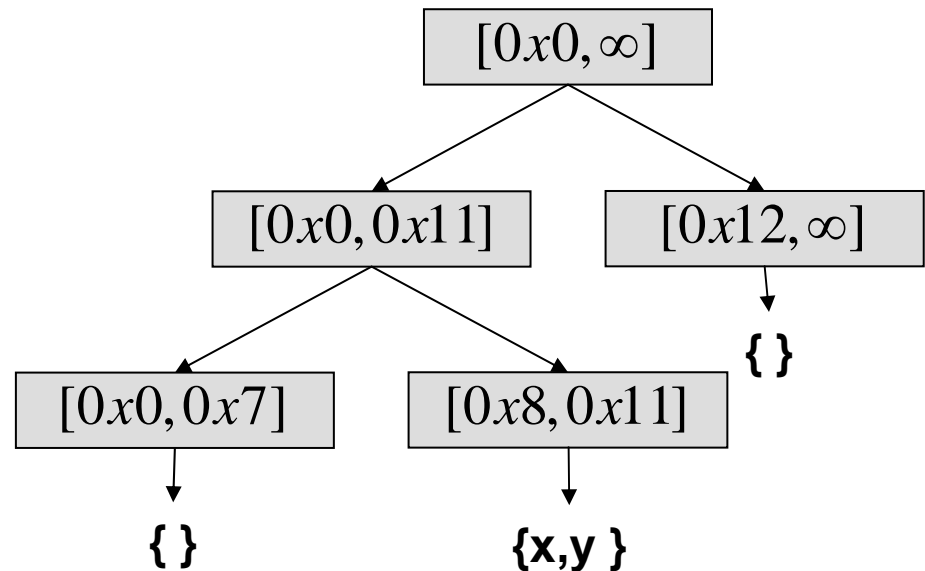
minimal slice

- Memory consists of many cells
- Partitioning into fixed cells is not possible
- Dynamic model of memory necessary



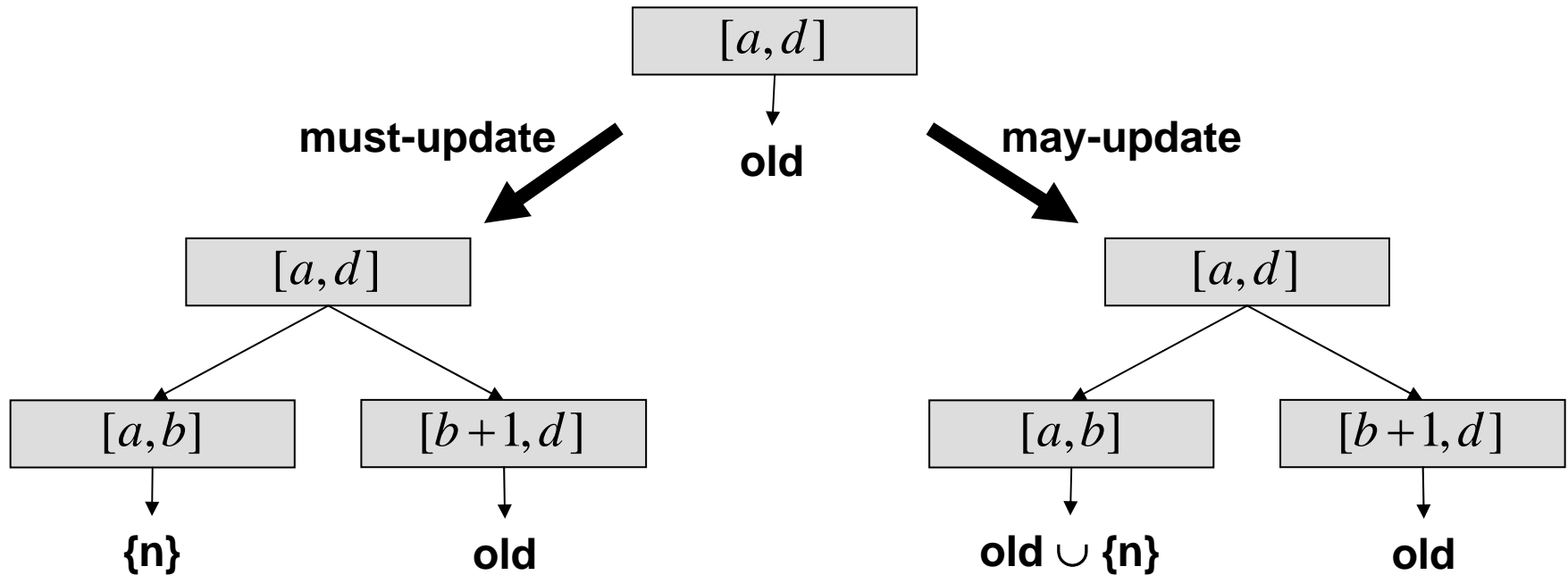
# Dynamic memory tree

- Binary tree
- Each inner node consists of an interval and two successors
- Each leaf consists of an interval and a set of program points
- Interval corresponds to bounds of different memory cells
- Start a forward analysis with  $([0x0, \infty], \{\})$



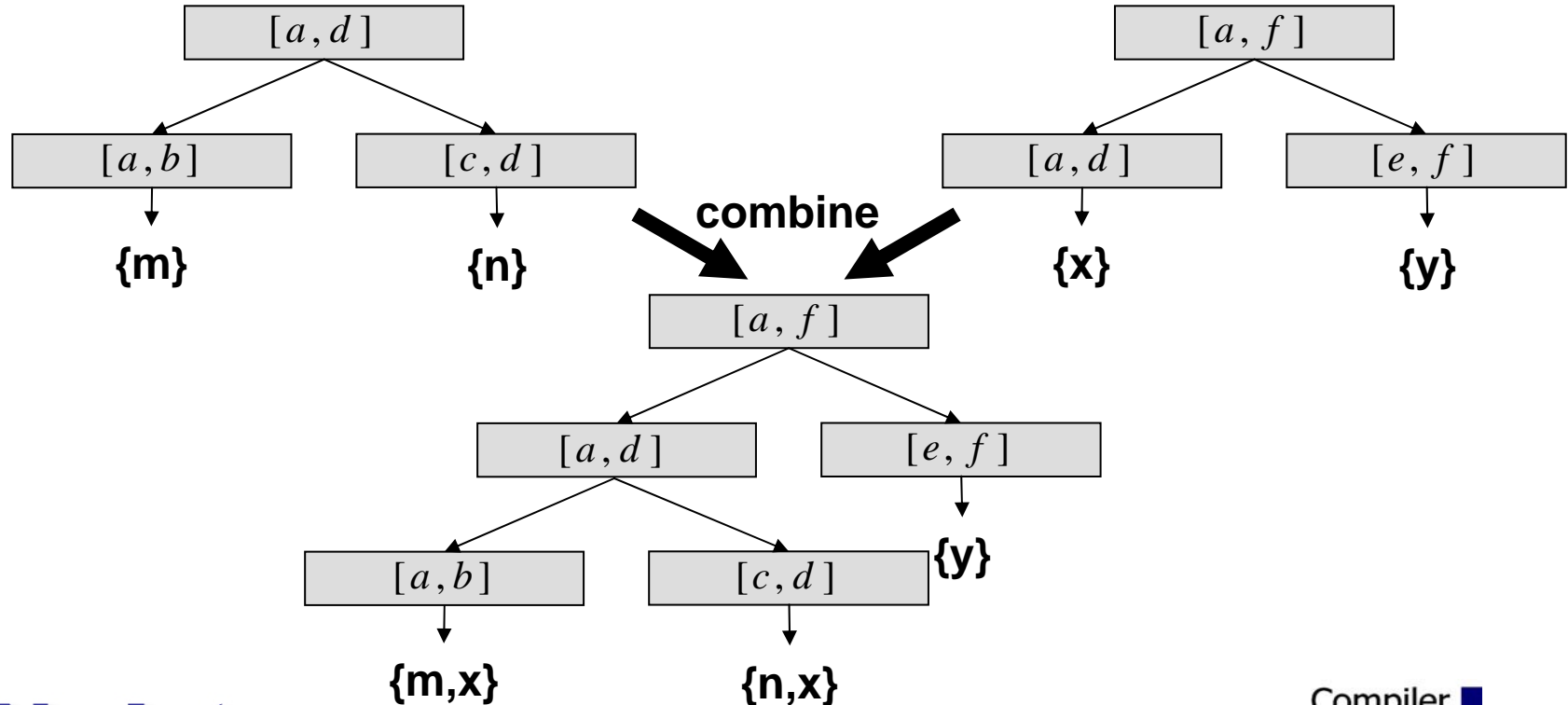
# Transfer function

- For instruction  $n$  writing to the memory cells  $[a,b]$



# Combine function

- Take the leafs of one tree and add it to the second one by doing a may-update



# Slicing the program (1)

---

- Already seen:
  - Reconstruction of data dependencies
  - Reconstruction of control dependencies
  - Dynamic model of the memory system
  
- Follows next:
  - Using these information to compute approximations to statement-minimal slices

# Slicing the program (2)

---

- Precomputation of data, control and memory dependencies
- while no abort
  - wait for slicing criterion  $(n, V)$
  - $\text{workset} = \text{Data}(n, V) \cup \text{Mem}(n) \cup \text{Data}(\text{Ctrl}(n)) \cup \text{Mem}(\text{Ctrl}(n))$
  - $\text{visited} = \{(n, V)\} \cup \text{Ctrl}(n)$
  - while  $\text{workset} \neq \{ \}$ 
    - $(x, W) \leftarrow \text{workset}$
    - $\text{visited} = \text{visited} \cup \{(x, W)\} \cup \text{Ctrl}(x)$
    - $\text{workset} = \text{workset} \setminus \{(x, W)\} \cup \{ \text{Data}(x, W) \cup \text{Mem}(x, W) \cup \text{Data}(\text{Ctrl}(x)) \cup \text{Mem}(\text{Ctrl}(x)) \setminus \text{visited} \}$
  - $\text{slice} = \{ n \mid (n, \_) \in \text{visited} \}$

# Experimental results (1)

---

- Usability depends on
  - Time for analyzing the executable
    - Depends on the analyzed executable
      - #instructions, #calls, #memory accesses
  - Time for computing a slice
    - Depends on the chosen criterion
  - Preciseness of calculated slices
    - Also depends on the chosen slicing criterion
    - Hard to verify

# Experimental results (2)

---

- Characteristics of some test files

	minmax	fac	prime	dry2_1	st30
routines	4	2	4	17	163
calls	5	2	4	32	309
instructions	114	24	119	773	3820
loops	0	1	2	9	50
loads	4	2	20	296	984
stores	4	2	10	140	877

# Experimental results (3)

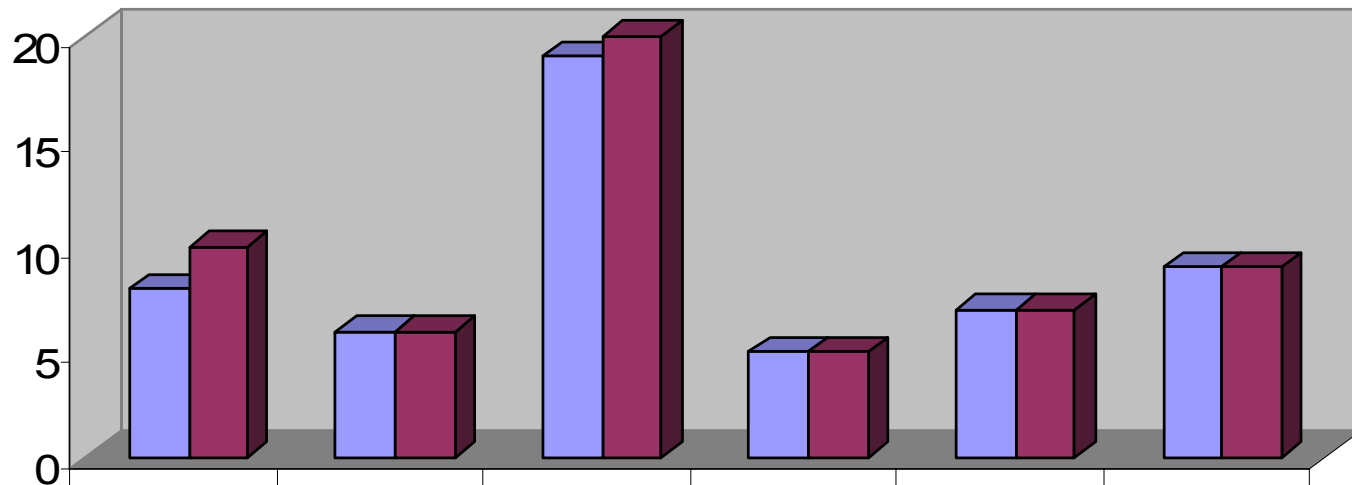
- Measured precomputation times (in seconds)

	minmax	fac	prime	dry2_1	st30
cfg	0.03	0.02	0.03	0.17	0.5
data	0.02	0.03	0.04	0.25	173.2
cp	0.01	0.02	0.02	0.1	0.4
pdom	0.01	0.01	0.01	0.07	0.4
mem	0.01	0.02	0.02	0.32	1.2
$\Sigma$	0.07	0.1	0.13	0.82	177.5



# Experimental results (4)

Variation from statement-minimal slices



	minmax criterion 1	minmax criterion 2	minmax criterion 3	fac criterion 1	fac criterion 2	fac criterion 3
minimal	8	6	19	5	7	9
computed	10	6	20	5	7	9

# Conclusions

---

- Generic solution for the slicing of executable programs
  - Only a few architecture specific parameters required
    - Registers
    - Register aliases
- Dynamic model of the memory
- Currently support for ARM, PPC5xx and PPC755 Architectures
- Successfully integrated in the aiT framework
  - Interactive visualization with aiSee3

The screenshot shows the aiT StackAnalyzer ARM7 IDE interface. The title bar indicates the file path: /home/schlickling/opt/share/aiT\_ARM7/examples/minmax/minmax.apf. The interface is divided into several panes:

- Files:** Shows the executable file aiT\_ARM7/examples/minmax/minmax.out and its start address arm::\_main. It also lists supporting files for AIP and AIS.
- Source:** A large empty area for source code, with a status bar at the bottom indicating "Line: 1, Column: 1".
- Disassembly:** Shows the disassembly of the `_main` function. The code is as follows:

```
.entry _main
.routine _main
arm:0x128: stmdb sp!, {lr}
arm:0x12c: sub sp, sp, #0xc
arm:0x130: mov r12, #0xa
arm:0x134: str r12, [sp]
arm:0x138: mov r12, #2
arm:0x13c: str r12, [sp, #+4]
arm:0x140: mov r12, #1
arm:0x144: str r12, [sp, #+8]
arm:0x148: ldr r0, [sp, #+4]
arm:0x14c: ldr r12, [sp]
arm:0x150: cmp r12, r0, LSL #0
arm:0x154: bgt 0x168 <0x168>
arm:0x158: mov r0, sp, LSL #0
arm:0x15c: add r1, sp, #4
arm:0x160: bl 0x20 <_swap>
```
- Messages:** Displays a warning message: "armdaan: Warning: loop analysis disabled (VIVU 4 Mapping necessary)".

aiT/StackAnalyzer ARM7: /home/schlickling/opt/share/aiT\_ARM7/examples/minmax/minmax.apf

File Actions Options Window Help

aiSee: /tmp/5291-minmax-arm\_\_main.gdl

Files

Executable

Start at: [ar]

Supporting F

AIP File: [ ]

AIS File: [7/e]

Disassembl

Function: [ \_m

```

.entry_main
.routine_ma
arm: 0x128:
arm: 0x12c:
arm: 0x130:
arm: 0x134:
arm: 0x138:
arm: 0x13c:
arm: 0x140:
arm: 0x144:
arm: 0x148:
arm: 0x14c:
arm: 0x150:
arm: 0x154:
arm: 0x158:
arm: 0x15c:
arm: 0x160:

```

The screenshot displays the aiT StackAnalyzer ARM7 interface. The main window shows a control flow graph (CFG) with several basic blocks. A dialog box titled "Select Context and Resources" is open, showing the instruction "str r1, [sp, #+4]". The dialog has two columns: "Contexts" and "Resources". Under "Contexts", "single context" is selected. Under "Resources", "isExecuted", "MEM", "r1", and "r13" are listed. The "Ok" and "Cancel" buttons are visible at the bottom of the dialog.

aiT/StackAnalyzer ARM7: /home/schlickling/opt/share/aiT\_ARM7/examples/minmax/minmax.apf

File Actions Options Window Help

aiSee: /tmp/5291-minmax-arm\_\_main.gdl

File View Options Help

aiSee: /tmp/5291-minmax-arm\_\_main.gdl

File View Options Help

Disassemble

Function: \_main

```

.entry_main
.routine_main
arm: 0x128:
arm: 0x12c:
arm: 0x130:
arm: 0x134:
arm: 0x138:
arm: 0x13c:
arm: 0x140:
arm: 0x144:
arm: 0x148:
arm: 0x14c:
arm: 0x150:
arm: 0x154:
arm: 0x158:
arm: 0x15c:
arm: 0x160:

```

The control flow graph consists of the following nodes (addresses in ARM7):

- 0x128: push sp, #8
- 0x12c: mov r12, #0x0
- 0x130: str r12, [sp]
- 0x134: mov r12, #2
- 0x138: str r12, [sp, #4]
- 0x13c: mov r12, #1
- 0x140: str r12, [sp, #8]
- 0x144: ldr r0, [sp, #4]
- 0x148: ldr r12, [sp]
- 0x14c: cmp r12, r0, LSL #0
- 0x150: bgt 0x158 <0x158>
- 0x154: arm: 0x158
- 0x158: arm: 0x154
- 0x15c: arm: 0x158
- 0x160: arm: 0x15c
- 0x164: arm: 0x160
- 0x168: arm: 0x164
- 0x16c: arm: 0x168
- 0x170: arm: 0x16c
- 0x174: arm: 0x170
- 0x178: arm: 0x174
- 0x184: arm: 0x178
- 0x188: arm: 0x184
- 0x194: arm: 0x188
- 0x198: arm: 0x194
- 0x1a4: arm: 0x198
- 0x1a8: arm: 0x1a4
- 0x1b4: arm: 0x1a8
- 0x1b8: arm: 0x1b4
- 0x1c4: arm: 0x1b8
- 0x1c8: arm: 0x1c4
- 0x1d4: arm: 0x1c8
- 0x1d8: arm: 0x1d4
- 0x1e0: arm: 0x1d8
- 0x1e4: arm: 0x1e0
- 0x1e8: arm: 0x1e4
- 0x1ec: arm: 0x1e8
- 0x1f0: arm: 0x1ec
- 0x1f4: arm: 0x1f0
- 0x1f8: arm: 0x1f4
- 0x204: arm: 0x1f8
- 0x208: arm: 0x204
- 0x214: arm: 0x208
- 0x218: arm: 0x214
- 0x224: arm: 0x218
- 0x228: arm: 0x224
- 0x234: arm: 0x228
- 0x238: arm: 0x234
- 0x244: arm: 0x238
- 0x248: arm: 0x244
- 0x254: arm: 0x248
- 0x258: arm: 0x254
- 0x264: arm: 0x258
- 0x268: arm: 0x264
- 0x274: arm: 0x268
- 0x278: arm: 0x274
- 0x284: arm: 0x278
- 0x288: arm: 0x284
- 0x294: arm: 0x288
- 0x298: arm: 0x294
- 0x2a4: arm: 0x298
- 0x2a8: arm: 0x2a4
- 0x2b4: arm: 0x2a8
- 0x2b8: arm: 0x2b4
- 0x2c4: arm: 0x2b8
- 0x2c8: arm: 0x2c4
- 0x2d4: arm: 0x2c8
- 0x2d8: arm: 0x2d4
- 0x2e4: arm: 0x2d8
- 0x2e8: arm: 0x2e4
- 0x2f4: arm: 0x2e8
- 0x2f8: arm: 0x2f4
- 0x304: arm: 0x2f8
- 0x308: arm: 0x304
- 0x314: arm: 0x308
- 0x318: arm: 0x314
- 0x324: arm: 0x318
- 0x328: arm: 0x324
- 0x334: arm: 0x328
- 0x338: arm: 0x334
- 0x344: arm: 0x338
- 0x348: arm: 0x344
- 0x354: arm: 0x348
- 0x358: arm: 0x354
- 0x364: arm: 0x358
- 0x368: arm: 0x364
- 0x374: arm: 0x368
- 0x378: arm: 0x374
- 0x384: arm: 0x378
- 0x388: arm: 0x384
- 0x394: arm: 0x388
- 0x398: arm: 0x394
- 0x3a4: arm: 0x398
- 0x3a8: arm: 0x3a4
- 0x3b4: arm: 0x3a8
- 0x3b8: arm: 0x3b4
- 0x3c4: arm: 0x3b8
- 0x3c8: arm: 0x3c4
- 0x3d4: arm: 0x3c8
- 0x3d8: arm: 0x3d4
- 0x3e4: arm: 0x3d8
- 0x3e8: arm: 0x3e4
- 0x3f4: arm: 0x3e8
- 0x3f8: arm: 0x3f4
- 0x404: arm: 0x3f8
- 0x408: arm: 0x404
- 0x414: arm: 0x408
- 0x418: arm: 0x414
- 0x424: arm: 0x418
- 0x428: arm: 0x424
- 0x434: arm: 0x428
- 0x438: arm: 0x434
- 0x444: arm: 0x438
- 0x448: arm: 0x444
- 0x454: arm: 0x448
- 0x458: arm: 0x454
- 0x464: arm: 0x458
- 0x468: arm: 0x464
- 0x474: arm: 0x468
- 0x478: arm: 0x474
- 0x484: arm: 0x478
- 0x488: arm: 0x484
- 0x494: arm: 0x488
- 0x498: arm: 0x494
- 0x4a4: arm: 0x498
- 0x4a8: arm: 0x4a4
- 0x4b4: arm: 0x4a8
- 0x4b8: arm: 0x4b4
- 0x4c4: arm: 0x4b8
- 0x4c8: arm: 0x4c4
- 0x4d4: arm: 0x4c8
- 0x4d8: arm: 0x4d4
- 0x4e4: arm: 0x4d8
- 0x4e8: arm: 0x4e4
- 0x4f4: arm: 0x4e8
- 0x4f8: arm: 0x4f4
- 0x504: arm: 0x4f8
- 0x508: arm: 0x504
- 0x514: arm: 0x508
- 0x518: arm: 0x514
- 0x524: arm: 0x518
- 0x528: arm: 0x524
- 0x534: arm: 0x528
- 0x538: arm: 0x534
- 0x544: arm: 0x538
- 0x548: arm: 0x544
- 0x554: arm: 0x548
- 0x558: arm: 0x554
- 0x564: arm: 0x558
- 0x568: arm: 0x564
- 0x574: arm: 0x568
- 0x578: arm: 0x574
- 0x584: arm: 0x578
- 0x588: arm: 0x584
- 0x594: arm: 0x588
- 0x598: arm: 0x594
- 0x5a4: arm: 0x598
- 0x5a8: arm: 0x5a4
- 0x5b4: arm: 0x5a8
- 0x5b8: arm: 0x5b4
- 0x5c4: arm: 0x5b8
- 0x5c8: arm: 0x5c4
- 0x5d4: arm: 0x5c8
- 0x5d8: arm: 0x5d4
- 0x5e4: arm: 0x5d8
- 0x5e8: arm: 0x5e4
- 0x5f4: arm: 0x5e8
- 0x5f8: arm: 0x5f4
- 0x604: arm: 0x5f8
- 0x608: arm: 0x604
- 0x614: arm: 0x608
- 0x618: arm: 0x614
- 0x624: arm: 0x618
- 0x628: arm: 0x624
- 0x634: arm: 0x628
- 0x638: arm: 0x634
- 0x644: arm: 0x638
- 0x648: arm: 0x644
- 0x654: arm: 0x648
- 0x658: arm: 0x654
- 0x664: arm: 0x658
- 0x668: arm: 0x664
- 0x674: arm: 0x668
- 0x678: arm: 0x674
- 0x684: arm: 0x678
- 0x688: arm: 0x684
- 0x694: arm: 0x688
- 0x698: arm: 0x694
- 0x6a4: arm: 0x698
- 0x6a8: arm: 0x6a4
- 0x6b4: arm: 0x6a8
- 0x6b8: arm: 0x6b4
- 0x6c4: arm: 0x6b8
- 0x6c8: arm: 0x6c4
- 0x6d4: arm: 0x6c8
- 0x6d8: arm: 0x6d4
- 0x6e4: arm: 0x6d8
- 0x6e8: arm: 0x6e4
- 0x6f4: arm: 0x6e8
- 0x6f8: arm: 0x6f4
- 0x704: arm: 0x6f8
- 0x708: arm: 0x704
- 0x714: arm: 0x708
- 0x718: arm: 0x714
- 0x724: arm: 0x718
- 0x728: arm: 0x724
- 0x734: arm: 0x728
- 0x738: arm: 0x734
- 0x744: arm: 0x738
- 0x748: arm: 0x744
- 0x754: arm: 0x748
- 0x758: arm: 0x754
- 0x764: arm: 0x758
- 0x768: arm: 0x764
- 0x774: arm: 0x768
- 0x778: arm: 0x774
- 0x784: arm: 0x778
- 0x788: arm: 0x784
- 0x794: arm: 0x788
- 0x798: arm: 0x794
- 0x7a4: arm: 0x798
- 0x7a8: arm: 0x7a4
- 0x7b4: arm: 0x7a8
- 0x7b8: arm: 0x7b4
- 0x7c4: arm: 0x7b8
- 0x7c8: arm: 0x7c4
- 0x7d4: arm: 0x7c8
- 0x7d8: arm: 0x7d4
- 0x7e4: arm: 0x7d8
- 0x7e8: arm: 0x7e4
- 0x7f4: arm: 0x7e8
- 0x7f8: arm: 0x7f4
- 0x804: arm: 0x7f8
- 0x808: arm: 0x804
- 0x814: arm: 0x808
- 0x818: arm: 0x814
- 0x824: arm: 0x818
- 0x828: arm: 0x824
- 0x834: arm: 0x828
- 0x838: arm: 0x834
- 0x844: arm: 0x838
- 0x848: arm: 0x844
- 0x854: arm: 0x848
- 0x858: arm: 0x854
- 0x864: arm: 0x858
- 0x868: arm: 0x864
- 0x874: arm: 0x868
- 0x878: arm: 0x874
- 0x884: arm: 0x878
- 0x888: arm: 0x884
- 0x894: arm: 0x888
- 0x898: arm: 0x894
- 0x8a4: arm: 0x898
- 0x8a8: arm: 0x8a4
- 0x8b4: arm: 0x8a8
- 0x8b8: arm: 0x8b4
- 0x8c4: arm: 0x8b8
- 0x8c8: arm: 0x8c4
- 0x8d4: arm: 0x8c8
- 0x8d8: arm: 0x8d4
- 0x8e4: arm: 0x8d8
- 0x8e8: arm: 0x8e4
- 0x8f4: arm: 0x8e8
- 0x8f8: arm: 0x8f4
- 0x904: arm: 0x8f8
- 0x908: arm: 0x904
- 0x914: arm: 0x908
- 0x918: arm: 0x914
- 0x924: arm: 0x918
- 0x928: arm: 0x924
- 0x934: arm: 0x928
- 0x938: arm: 0x934
- 0x944: arm: 0x938
- 0x948: arm: 0x944
- 0x954: arm: 0x948
- 0x958: arm: 0x954
- 0x964: arm: 0x958
- 0x968: arm: 0x964
- 0x974: arm: 0x968
- 0x978: arm: 0x974
- 0x984: arm: 0x978
- 0x988: arm: 0x984
- 0x994: arm: 0x988
- 0x998: arm: 0x994
- 0x9a4: arm: 0x998
- 0x9a8: arm: 0x9a4
- 0x9b4: arm: 0x9a8
- 0x9b8: arm: 0x9b4
- 0x9c4: arm: 0x9b8
- 0x9c8: arm: 0x9c4
- 0x9d4: arm: 0x9c8
- 0x9d8: arm: 0x9d4
- 0x9e4: arm: 0x9d8
- 0x9e8: arm: 0x9e4
- 0x9f4: arm: 0x9e8
- 0x9f8: arm: 0x9f4
- 0xa04: arm: 0x9f8
- 0xa08: arm: 0xa04
- 0xa14: arm: 0xa08
- 0xa18: arm: 0xa14
- 0xa24: arm: 0xa18
- 0xa28: arm: 0xa24
- 0xa34: arm: 0xa28
- 0xa38: arm: 0xa34
- 0xa44: arm: 0xa38
- 0xa48: arm: 0xa44
- 0xa54: arm: 0xa48
- 0xa58: arm: 0xa54
- 0xa64: arm: 0xa58
- 0xa68: arm: 0xa64
- 0xa74: arm: 0xa68
- 0xa78: arm: 0xa74
- 0xa84: arm: 0xa78
- 0xa88: arm: 0xa84
- 0xa94: arm: 0xa88
- 0xa98: arm: 0xa94
- 0xaa4: arm: 0xa98
- 0xaa8: arm: 0xaa4
- 0xab4: arm: 0xaa8
- 0xab8: arm: 0xab4
- 0xac4: arm: 0xab8
- 0xac8: arm: 0xac4
- 0xad4: arm: 0xac8
- 0xad8: arm: 0xad4
- 0xae4: arm: 0xad8
- 0xae8: arm: 0xae4
- 0xaf4: arm: 0xae8
- 0xaf8: arm: 0xaf4
- 0xb04: arm: 0xaf8
- 0xb08: arm: 0xb04
- 0xb14: arm: 0xb08
- 0xb18: arm: 0xb14
- 0xb24: arm: 0xb18
- 0xb28: arm: 0xb24
- 0xb34: arm: 0xb28
- 0xb38: arm: 0xb34
- 0xb44: arm: 0xb38
- 0xb48: arm: 0xb44
- 0xb54: arm: 0xb48
- 0xb58: arm: 0xb54
- 0xb64: arm: 0xb58
- 0xb68: arm: 0xb64
- 0xb74: arm: 0xb68
- 0xb78: arm: 0xb74
- 0xb84: arm: 0xb78
- 0xb88: arm: 0xb84
- 0xb94: arm: 0xb88
- 0xb98: arm: 0xb94
- 0xba4: arm: 0xb98
- 0xba8: arm: 0xba4
- 0xbb4: arm: 0xba8
- 0xbb8: arm: 0xbb4
- 0xbc4: arm: 0xbb8
- 0xbc8: arm: 0xbc4
- 0xbd4: arm: 0xbc8
- 0xbd8: arm: 0xbd4
- 0xbe4: arm: 0xbd8
- 0xbe8: arm: 0xbe4
- 0xbf4: arm: 0xbe8
- 0xbf8: arm: 0xbf4
- 0xc04: arm: 0xbf8
- 0xc08: arm: 0xc04
- 0xc14: arm: 0xc08
- 0xc18: arm: 0xc14
- 0xc24: arm: 0xc18
- 0xc28: arm: 0xc24
- 0xc34: arm: 0xc28
- 0xc38: arm: 0xc34
- 0xc44: arm: 0xc38
- 0xc48: arm: 0xc44
- 0xc54: arm: 0xc48
- 0xc58: arm: 0xc54
- 0xc64: arm: 0xc58
- 0xc68: arm: 0xc64
- 0xc74: arm: 0xc68
- 0xc78: arm: 0xc74
- 0xc84: arm: 0xc78
- 0xc88: arm: 0xc84
- 0xc94: arm: 0xc88
- 0xc98: arm: 0xc94
- 0xca4: arm: 0xc98
- 0xca8: arm: 0xca4
- 0xcb4: arm: 0xca8
- 0xcb8: arm: 0xcb4
- 0xcc4: arm: 0xcb8
- 0xcc8: arm: 0xcc4
- 0xcd4: arm: 0xcc8
- 0xcd8: arm: 0xcd4
- 0xce4: arm: 0xcd8
- 0xce8: arm: 0xce4
- 0xcf4: arm: 0xce8
- 0xcf8: arm: 0xcf4
- 0xd04: arm: 0xcf8
- 0xd08: arm: 0xd04
- 0xd14: arm: 0xd08
- 0xd18: arm: 0xd14
- 0xd24: arm: 0xd18
- 0xd28: arm: 0xd24
- 0xd34: arm: 0xd28
- 0xd38: arm: 0xd34
- 0xd44: arm: 0xd38
- 0xd48: arm: 0xd44
- 0xd54: arm: 0xd48
- 0xd58: arm: 0xd54
- 0xd64: arm: 0xd58
- 0xd68: arm: 0xd64
- 0xd74: arm: 0xd68
- 0xd78: arm: 0xd74
- 0xd84: arm: 0xd78
- 0xd88: arm: 0xd84
- 0xd94: arm: 0xd88
- 0xd98: arm: 0xd94
- 0xda4: arm: 0xd98
- 0xda8: arm: 0xda4
- 0xdb4: arm: 0xda8
- 0xdb8: arm: 0xdb4
- 0xdc4: arm: 0xdb8
- 0xdc8: arm: 0xdc4
- 0xdd4: arm: 0xdc8
- 0xdd8: arm: 0xdd4
- 0xde4: arm: 0xdd8
- 0xde8: arm: 0xde4
- 0xdf4: arm: 0xde8
- 0xdf8: arm: 0xdf4
- 0xe04: arm: 0xdf8
- 0xe08: arm: 0xe04
- 0xe14: arm: 0xe08
- 0xe18: arm: 0xe14
- 0xe24: arm: 0xe18
- 0xe28: arm: 0xe24
- 0xe34: arm: 0xe28
- 0xe38: arm: 0xe34
- 0xe44: arm: 0xe38
- 0xe48: arm: 0xe44
- 0xe54: arm: 0xe48
- 0xe58: arm: 0xe54
- 0xe64: arm: 0xe58
- 0xe68: arm: 0xe64
- 0xe74: arm: 0xe68
- 0xe78: arm: 0xe74
- 0xe84: arm: 0xe78
- 0xe88: arm: 0xe84
- 0xe94: arm: 0xe88
- 0xe98: arm: 0xe94
- 0xea4: arm: 0xe98
- 0xea8: arm: 0xea4
- 0xeb4: arm: 0xea8
- 0xeb8: arm: 0xeb4
- 0xec4: arm: 0xeb8
- 0xec8: arm: 0xec4
- 0xed4: arm: 0xec8
- 0xed8: arm: 0xed4
- 0xee4: arm: 0xed8
- 0xee8: arm: 0xee4
- 0xef4: arm: 0xee8
- 0xef8: arm: 0xef4
- 0xf04: arm: 0xef8
- 0xf08: arm: 0xf04
- 0xf14: arm: 0xf08
- 0xf18: arm: 0xf14
- 0xf24: arm: 0xf18
- 0xf28: arm: 0xf24
- 0xf34: arm: 0xf28
- 0xf38: arm: 0xf34
- 0xf44: arm: 0xf38
- 0xf48: arm: 0xf44
- 0xf54: arm: 0xf48
- 0xf58: arm: 0xf54
- 0xf64: arm: 0xf58
- 0xf68: arm: 0xf64
- 0xf74: arm: 0xf68
- 0xf78: arm: 0xf74
- 0xf84: arm: 0xf78
- 0xf88: arm: 0xf84
- 0xf94: arm: 0xf88
- 0xf98: arm: 0xf94
- 0xfa4: arm: 0xf98
- 0xfa8: arm: 0xfa4
- 0xfb4: arm: 0xfa8
- 0xfb8: arm: 0xfb4
- 0xfc4: arm: 0xfb8
- 0xfc8: arm: 0xfc4
- 0xfd4: arm: 0xfc8
- 0xfd8: arm: 0xfd4
- 0xfe4: arm: 0xfd8
- 0xfe8: arm: 0xfe4
- 0xff4: arm: 0xfe8
- 0xff8: arm: 0xff4

The graph shows a sequence of instructions with branches. A yellow highlight covers the main body of the function, starting from the first branch (0x158) and ending at the final return instruction (0x1e0). A smaller yellow box highlights the final return instruction (0x1e0).