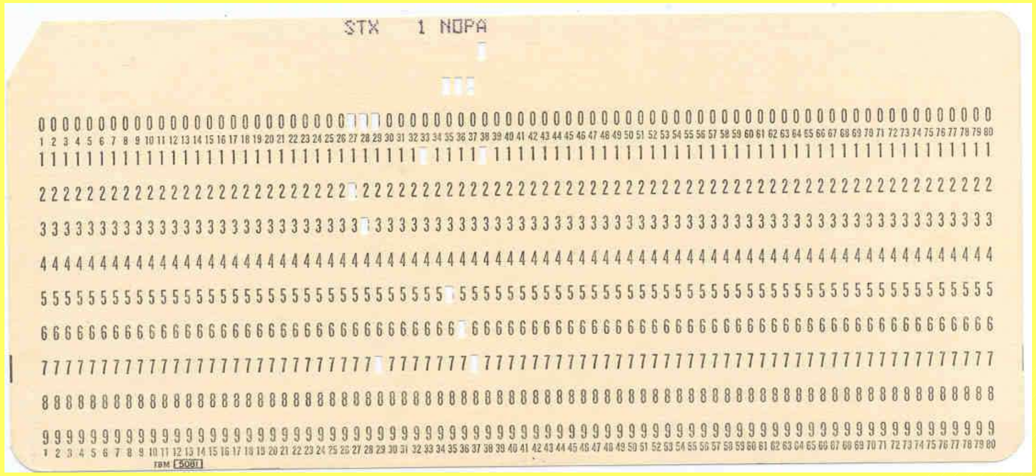
Lancet: A Nifty Code Editing Tool

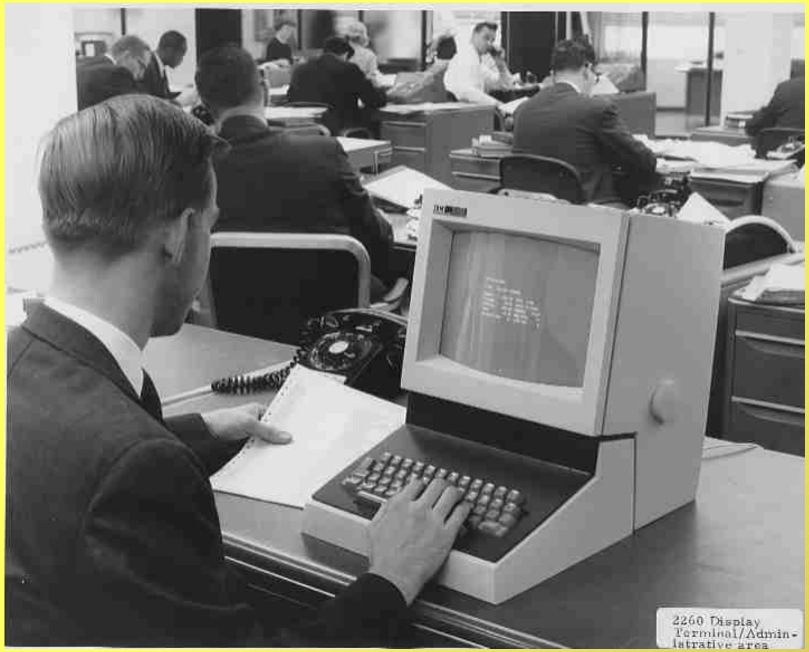
Ludo Van Put, Bjorn De Sutter, Matias Madou,

Bruno De Bus, Dominique Chanet,

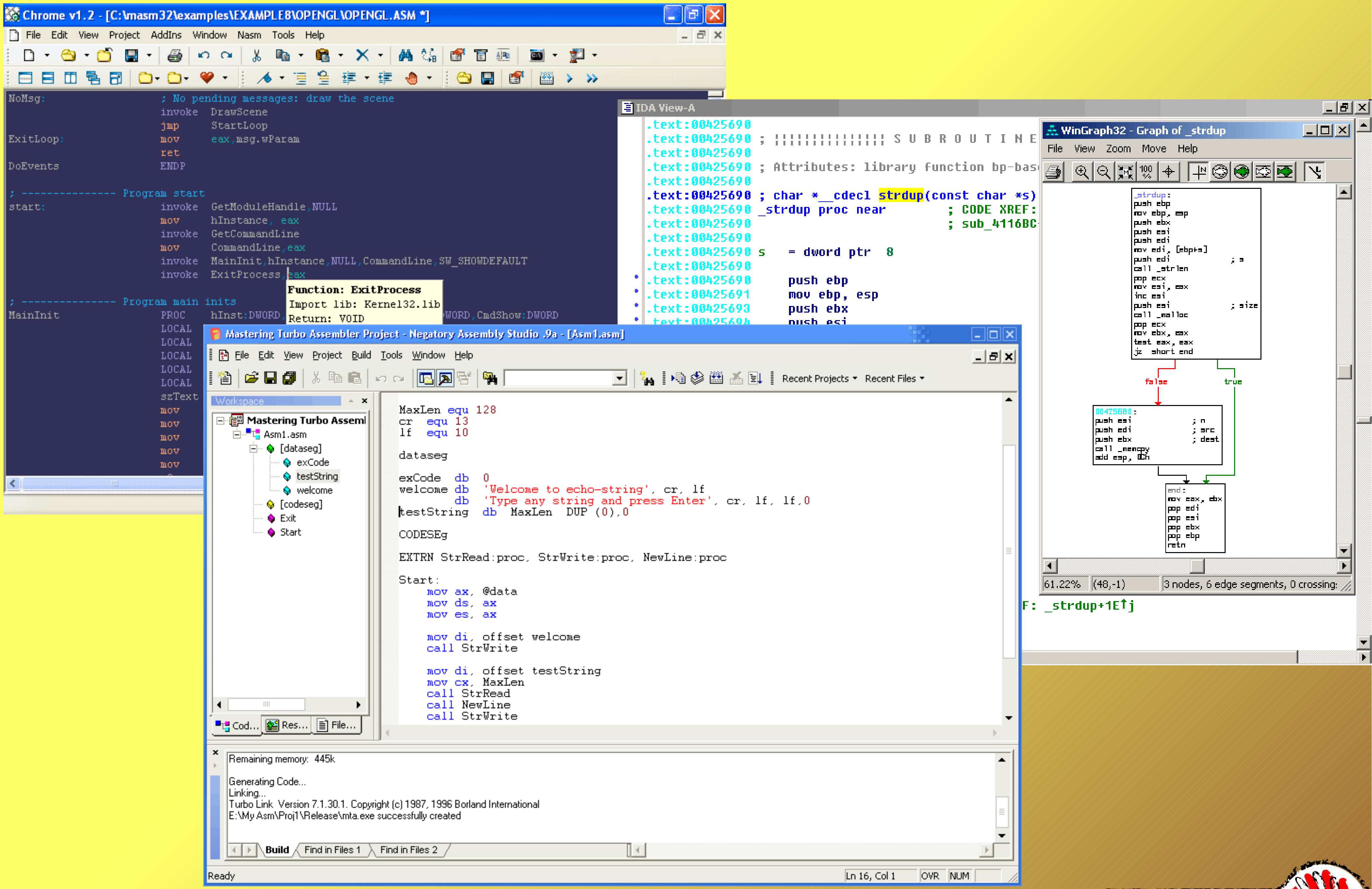
Kristof Smits & Koen De Bosschere

Assembly programming has evolved..





Assembly programming has evolved..


Chrome v1.2 - [C:\masm32\examples\EXAMPLE8\OPENGL\OPENGL.ASM *]
File Edit View Project AddIns Window Nasm Tools Help
IDA View-A
WinGraph32 - Graph of _strdup
File View Zoom Move Help
Mastering Turbo Assembler Project - Negatory Assembly Studio .9a - [Asm1.asm]
File Edit View Project Build Tools Window Help
Recent Projects
Recent Files
Workspace
Mastering Turbo Assembl
Asm1.asm
[dataseg]
exCode
testString
welcome
[codeseg]
Exit
Start
Cod...
Res...
File...
Build
Find in Files 1
Find in Files 2
Ready
Ln 16, Col 1
OVR
NUM
false
true

```
NoMsg:                  ; No pending messages: draw the scene
                        invoke  DrawScene
                        jmp     StartLoop
ExitLoop:               mov     eax,msg.wParam
                        ret
DoEvents                ENDP

; --------------- Program start
start:                  invoke  GetModuleHandle,NULL
                        mov     hInstance, eax
                        invoke  GetCommandLine
                        mov     CommandLine,eax
                        invoke  MainInit,hInstance,NULL,CommandLine,SW_SHOWDEFAULT
                        invoke  ExitProcess,eax

; --------------- Program main inits
MainInit                PROC    hInst:DWORD                 WORD,CmdShow:DWORD
                        LOCAL
                        LOCAL
                        LOCAL
                        LOCAL
                        LOCAL
                        szText
                        mov
                        mov
                        mov
                        mov
                        mov
```

```
Function: ExitProcess
Import lib: Kernel32.lib
Return: VOID
```

```
.text:00425690
.text:00425690 ; ||||||||||||||| S U B R O U T I N E
.text:00425690
.text:00425690 ; Attributes: library function bp-bas
.text:00425690
.text:00425690 ; char *__cdecl strdup(const char *s)
.text:00425690 _strdup proc near              ; CODE XREF:
.text:00425690                                ; sub_4116BC
.text:00425690
.text:00425690 s     = dword ptr  8
.text:00425690
.text:00425690       push ebp
.text:00425691       mov ebp, esp
.text:00425693       push ebx
.text:00425694       push esi
```

```
F: _strdup+1E↑j
```

```
_strdup:
push ebp
mov ebp, esp
push ebx
push esi
push edi
mov edi, [ebp+s]
push edi                ; s
call _strlen
pop ecx
mov esi, eax
inc esi
push esi                ; size
call _malloc
pop ecx
mov ebx, eax
test eax, eax
jz  short end
```

```
00425681:
push esi                ; n
push edi                ; src
push ebx                ; dest
call _memcpy
add esp, 0Ch
```

```
end:
mov eax, ebx
pop edi
pop esi
pop ebx
pop ebp
retn
```

```
61.22%   (48,-1)   3 nodes, 6 edge segments, 0 crossing:
```

```
MaxLen equ 128
cr   equ 13
lf   equ 10

dataseg

exCode  db   0
welcome db   'Welcome to echo-string', cr, lf
        db   'Type any string and press Enter', cr, lf, lf,0
testString  db  MaxLen  DUP (0),0

CODESEg

EXTRN StrRead:proc, StrWrite:proc, NewLine:proc

Start:
    mov ax, @data
    mov ds, ax
    mov es, ax

    mov di, offset welcome
    call StrWrite

    mov di, offset testString
    mov cx, MaxLen
    call StrRead
    call NewLine
    call StrWrite
```

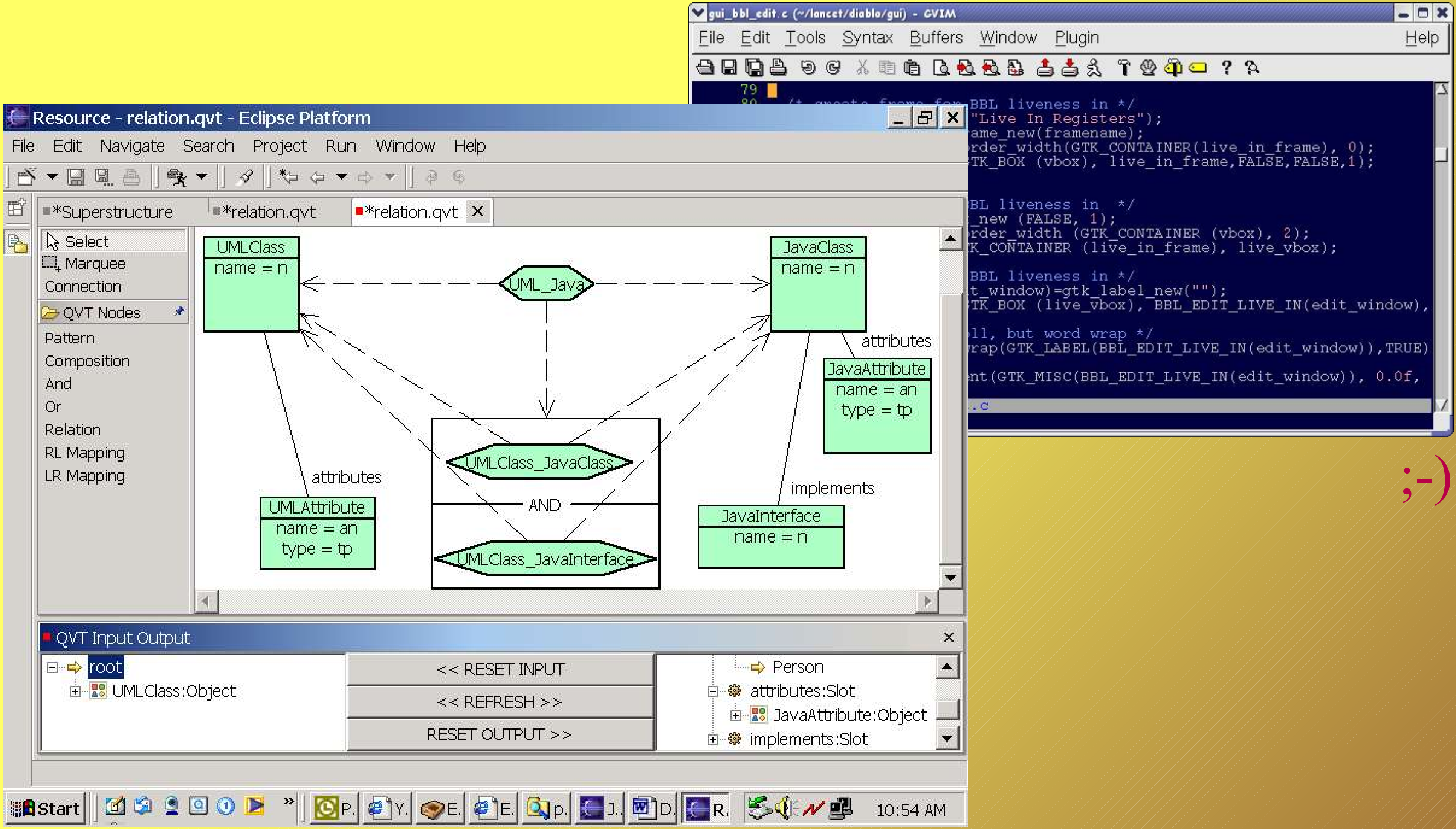
```
Remaining memory:  445k

Generating Code...
Linking...
Turbo Link  Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International
E:\My Asm\Proj1\Release\mta.exe successfully created
```

Lancet: A Nifty Code Editing Tool

Assembly programming has evolved..

But who cares when you can choose between a plethora of powerful software engineering tools?



```
BBL liveness in */
"Live In Registers");
ame_new(framename);
rder_width(GTK_CONTAINER(live_in_frame), 0);
TK_BOX (vbox), live_in_frame,FALSE,FALSE,1);

BL liveness in  */
_new (FALSE, 1);
rder_width (GTK_CONTAINER (vbox), 2);
K_CONTAINER (live_in_frame), live_vbox);

BBL liveness in */
t_window)=gtk_label_new("");
TK_BOX (live_vbox), BBL_EDIT_LIVE_IN(edit_window),

ll, but word wrap */
rap(GTK_LABEL(BBL_EDIT_LIVE_IN(edit_window)),TRUE)

nt(GTK_MISC(BBL_EDIT_LIVE_IN(edit_window)), 0.0f,
```

;-)

Program analysis and development

Assembly code is still written and/or analyzed by

- embedded systems developers
- device driver developers
- compiler writers
- computer science students
- ...

However, the available tools suffer from at least one of the following shortcomings:

- show only one level of abstraction
- linear list of instructions, control flow is unclear
- no program overview, calling context is unknown
- no feedback on changes made to the code
- ...

Overview

- the design of **Lancet**
- functionality
- user scenarios
- future work

Graphical user interface on top of Diablo

- **Lancet** is built on top of **Diablo** (http://www.elis.ugent.be/diablo)
- **Diablo** is an open source, retargetable link-time binary rewriting framework
- Spawned a program compaction tool (LCTES'04), a OS kernel compactor (LCTES'05), an instrumentation toolkit (PASTE'04), a steganography tool (ICISC'04)
- Offers a rich collection of program analyses and transformations to be reused in **Lancet**

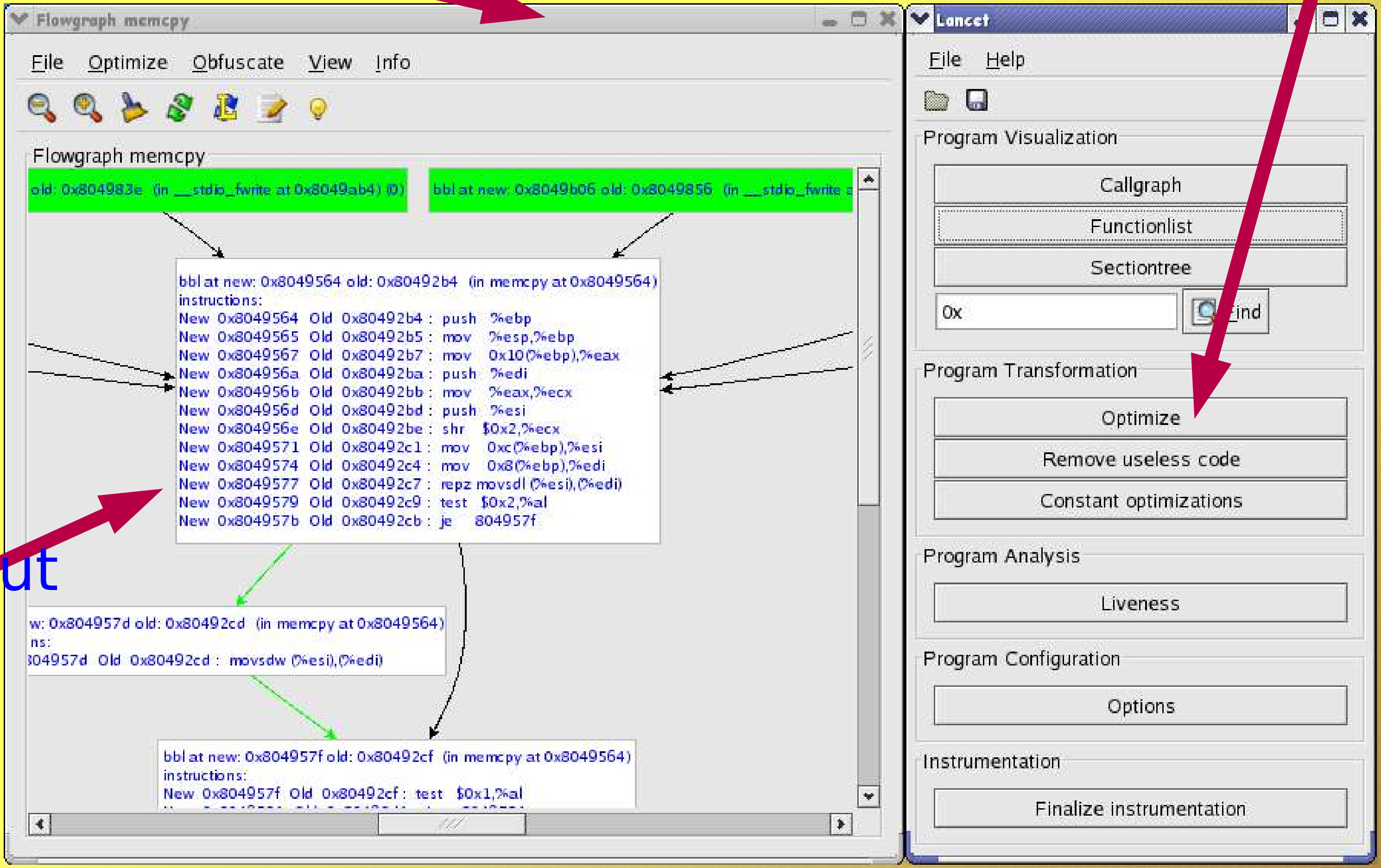
The design of Lancet

graphical interface

rewriting framework

graph layout

Graphviz



Flowgraph memcpy

```
old: 0x804983e  (in __stdio_fwrite at 0x8049ab4) (0)
```

```
bbl at new: 0x8049b06 old: 0x8049856  (in __stdio_fwrite a
```

```
bbl at new: 0x8049564 old: 0x80492b4  (in memcpy at 0x8049564)
instructions:
New  0x8049564  Old  0x80492b4 :  push   %ebp
New  0x8049565  Old  0x80492b5 :  mov    %esp,%ebp
New  0x8049567  Old  0x80492b7 :  mov    0x10(%ebp),%eax
New  0x804956a  Old  0x80492ba :  push   %edi
New  0x804956b  Old  0x80492bb :  mov    %eax,%ecx
New  0x804956d  Old  0x80492bd :  push   %esi
New  0x804956e  Old  0x80492be :  shr    $0x2,%ecx
New  0x8049571  Old  0x80492c1 :  mov    0xc(%ebp),%esi
New  0x8049574  Old  0x80492c4 :  mov    0x8(%ebp),%edi
New  0x8049577  Old  0x80492c7 :  repz movsdl (%esi),(%edi)
New  0x8049579  Old  0x80492c9 :  test   $0x2,%al
New  0x804957b  Old  0x80492cb :  je     804957f
```

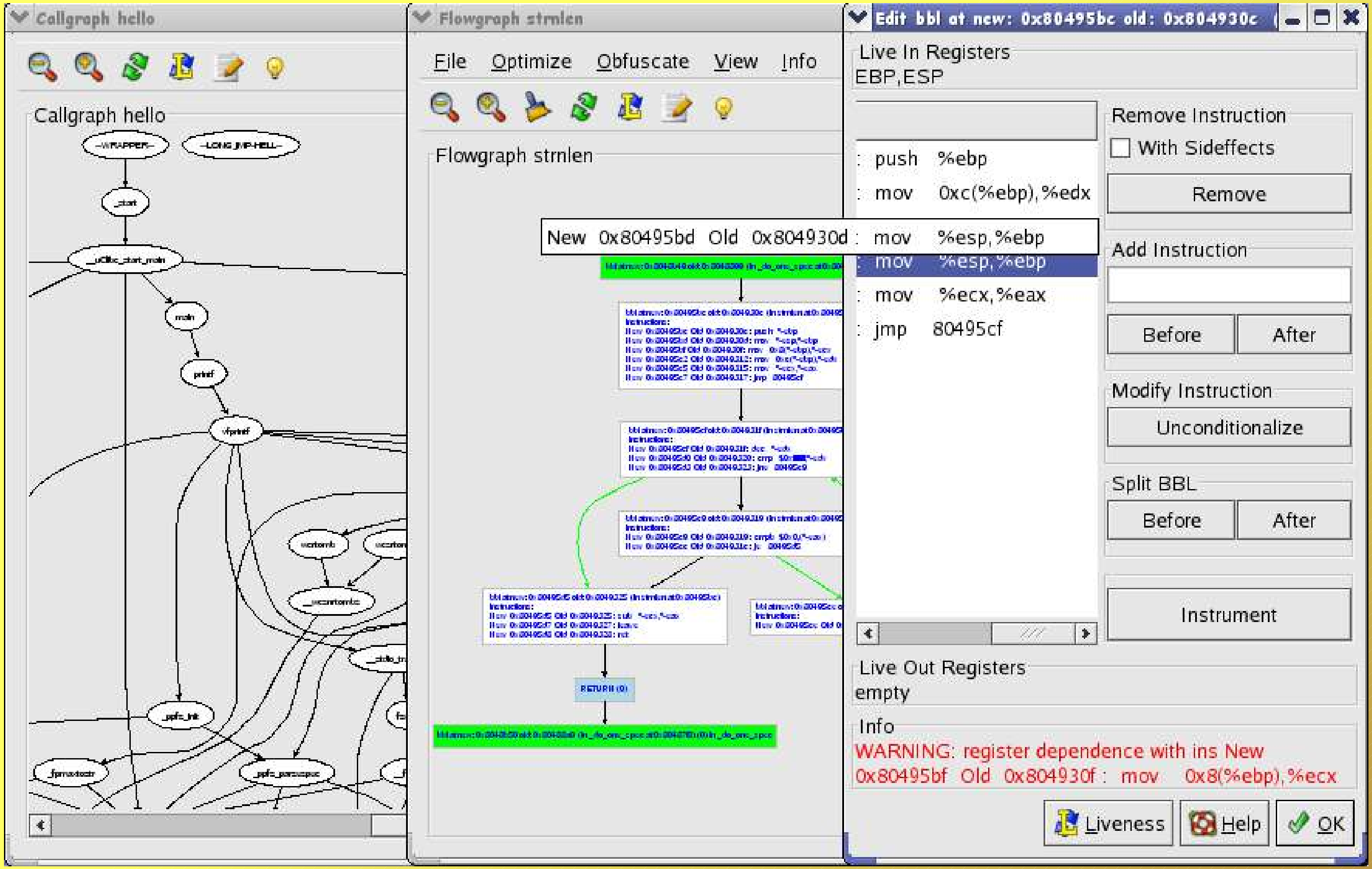
```
w: 0x804957d old: 0x80492cd  (in memcpy at 0x8049564)
ns:
304957d  Old  0x80492cd :  movsdw (%esi),(%edi)
```

```
bbl at new: 0x804957f old: 0x80492cf  (in memcpy at 0x8049564)
instructions:
New  0x804957f  Old  0x80492cf :  test   $0x1,%al
```

Functionality of Lancet

view different levels of abstraction

clear presentation of control flow



Live In Registers
EBP,ESP

```
: push   %ebp
: mov    0xc(%ebp),%edx
New 0x80495bd Old 0x804930d : mov   %esp,%ebp
: mov    %esp,%ebp
: mov    %ecx,%eax
: jmp    80495cf
```

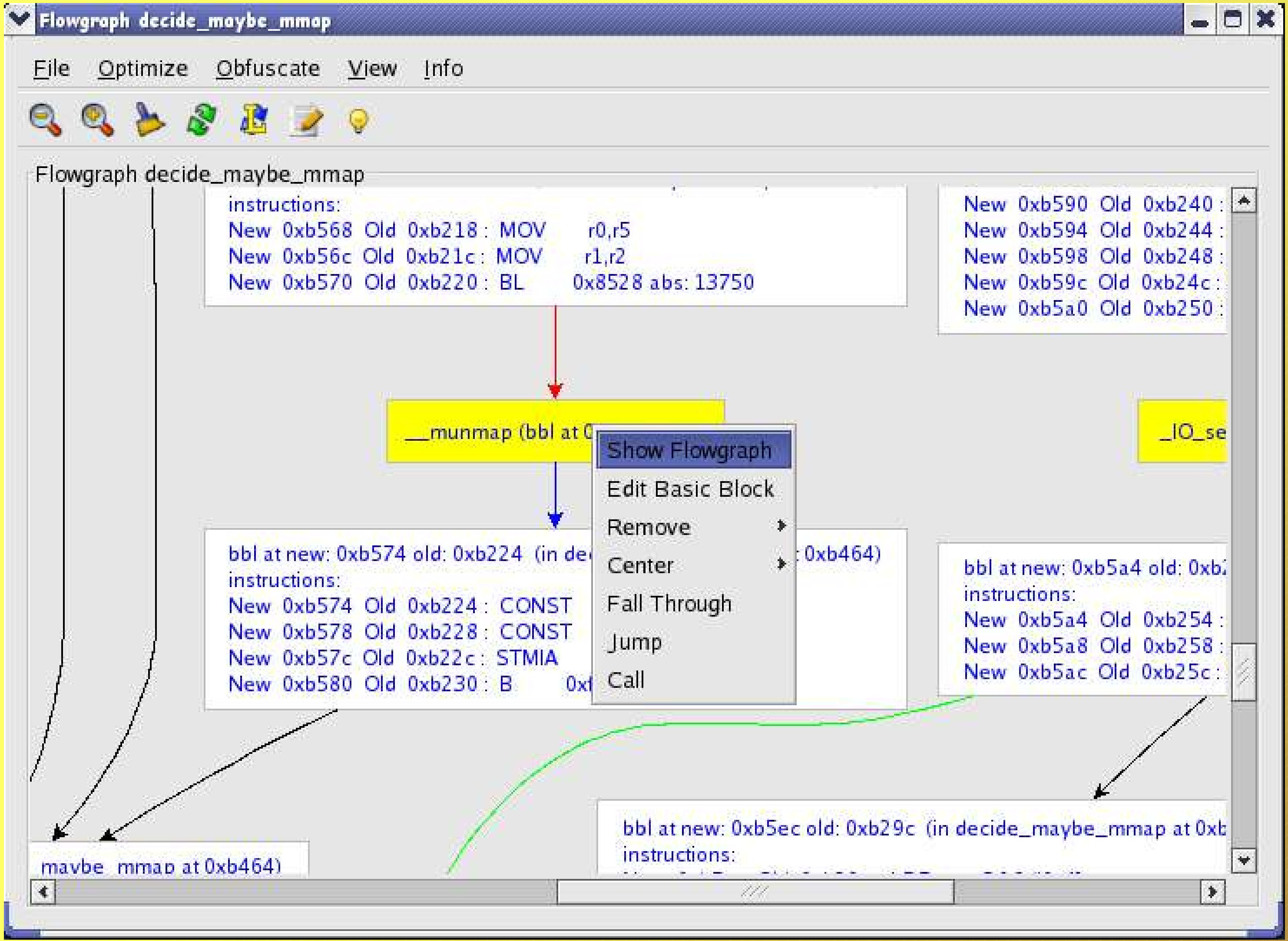
Live Out Registers
empty

Info
WARNING: register dependence with ins New 0x80495bf Old 0x804930f : mov 0x8(%ebp),%ecx

whole program overview

feedback from underlying framework

Functionality of Lancet



Flowgraph decide_maybe_mmap

```
instructions:
New 0xb568 Old 0xb218 : MOV     r0,r5
New 0xb56c Old 0xb21c : MOV     r1,r2
New 0xb570 Old 0xb220 : BL      0x8528 abs: 13750
```

```
New 0xb590 Old 0xb240 :
New 0xb594 Old 0xb244 :
New 0xb598 Old 0xb248 :
New 0xb59c Old 0xb24c :
New 0xb5a0 Old 0xb250 :
```

__munmap (bbl at 0

_IO_se

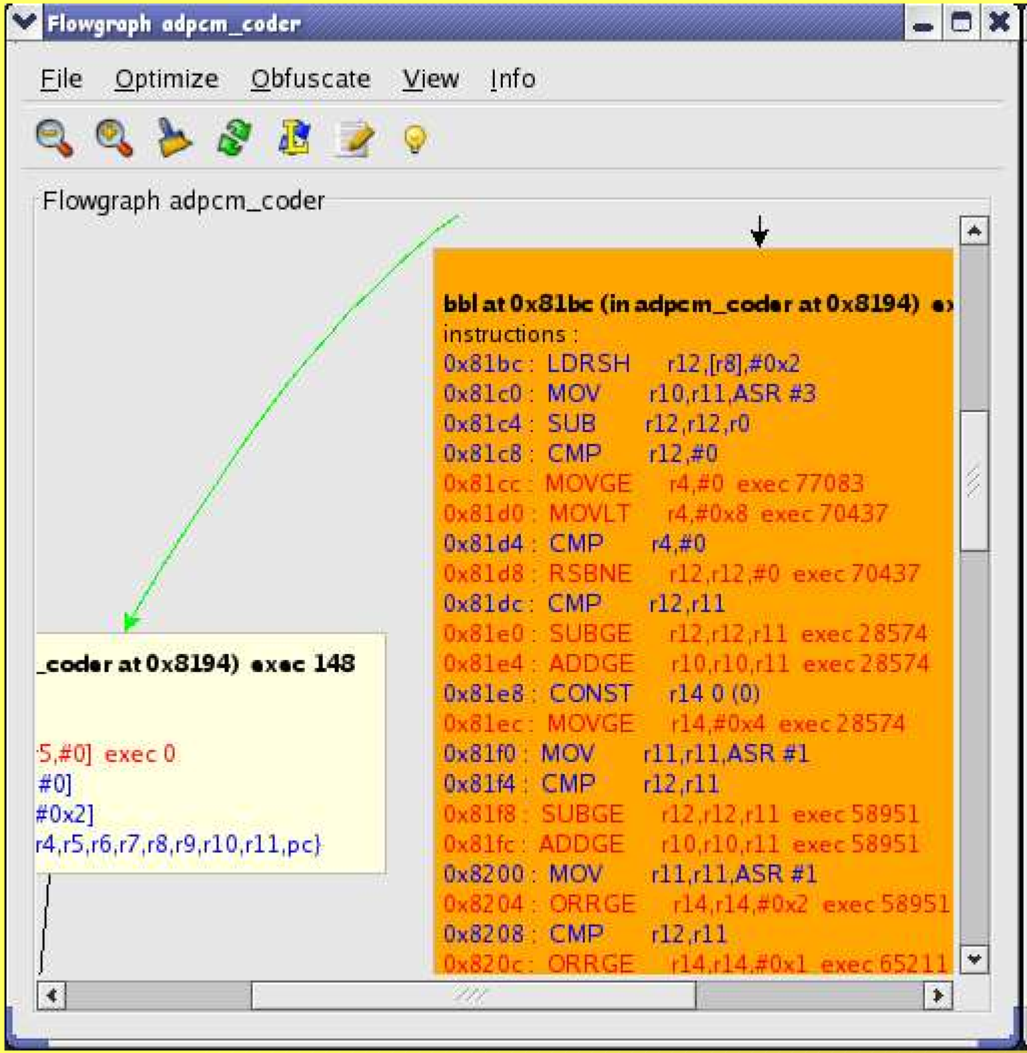
```
bbl at new: 0xb574 old: 0xb224 (in de           : 0xb464)
instructions:
New 0xb574 Old 0xb224 : CONST
New 0xb578 Old 0xb228 : CONST
New 0xb57c Old 0xb22c : STMIA
New 0xb580 Old 0xb230 : B       0xf
```

```
bbl at new: 0xb5a4 old: 0xb2
instructions:
New 0xb5a4 Old 0xb254 :
New 0xb5a8 Old 0xb258 :
New 0xb5ac Old 0xb25c :
```

maybe_mmap at 0xb464)

```
bbl at new: 0xb5ec old: 0xb29c (in decide_maybe_mmap at 0xb
instructions:
```

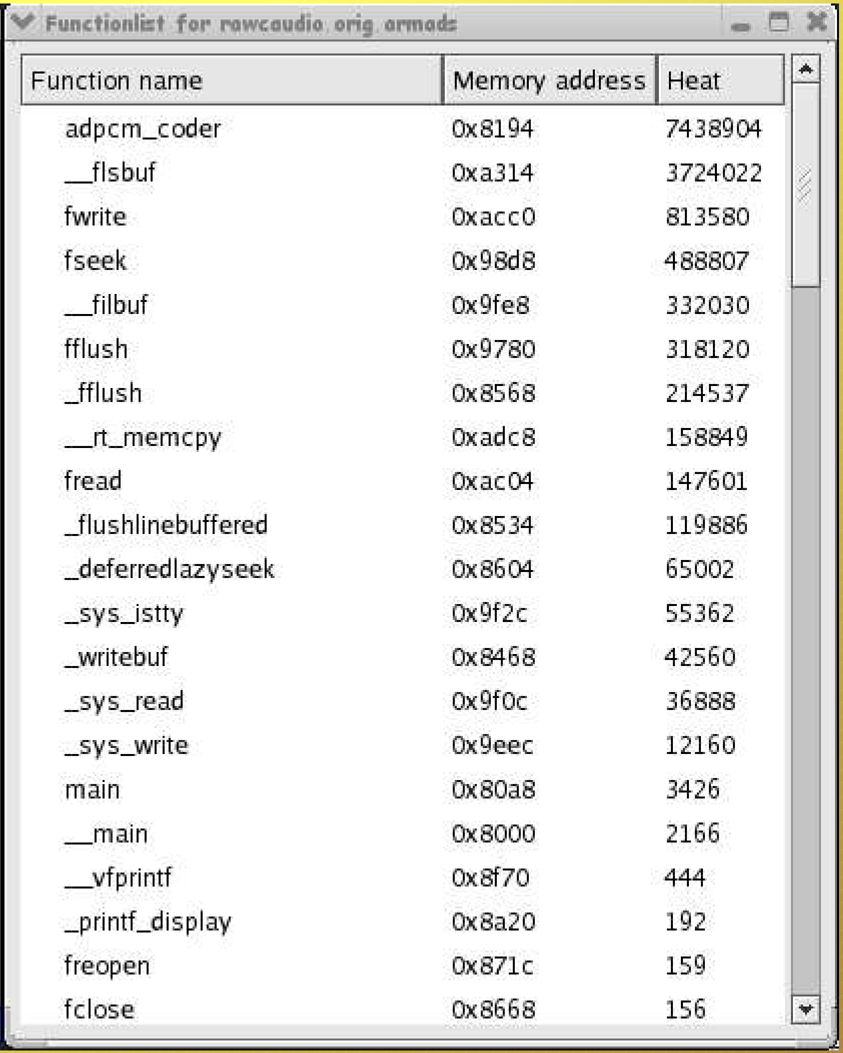
Functionality of Lancet



```
bbl at 0x81bc (in adpcm_coder at 0x8194) e
instructions :
0x81bc : LDRSH     r12,[r8],#0x2
0x81c0 : MOV       r10,r11,ASR #3
0x81c4 : SUB       r12,r12,r0
0x81c8 : CMP       r12,#0
0x81cc : MOVGE     r4,#0  exec 77083
0x81d0 : MOVLT     r4,#0x8  exec 70437
0x81d4 : CMP       r4,#0
0x81d8 : RSBNE     r12,r12,#0  exec 70437
0x81dc : CMP       r12,r11
0x81e0 : SUBGE     r12,r12,r11  exec 28574
0x81e4 : ADDGE     r10,r10,r11  exec 28574
0x81e8 : CONST     r14 0 (0)
0x81ec : MOVGE     r14,#0x4  exec 28574
0x81f0 : MOV       r11,r11,ASR #1
0x81f4 : CMP       r12,r11
0x81f8 : SUBGE     r12,r12,r11  exec 58951
0x81fc : ADDGE     r10,r10,r11  exec 58951
0x8200 : MOV       r11,r11,ASR #1
0x8204 : ORRGE     r14,r14,#0x2  exec 58951
0x8208 : CMP       r12,r11
0x820c : ORRGE     r14,r14,#0x1  exec 65211
```

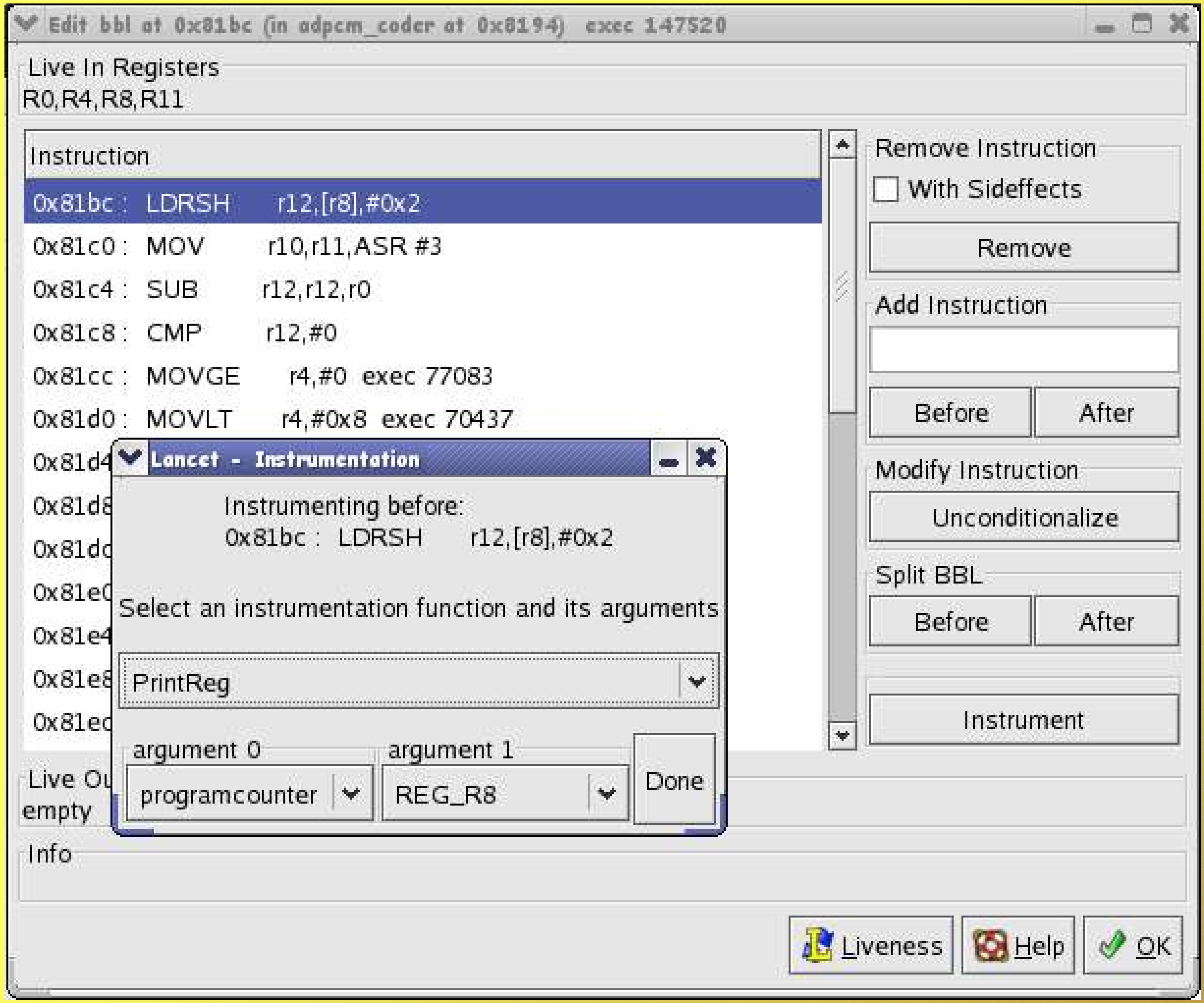
```
_coder at 0x8194) exec 148

5,#0]  exec 0
#0]
#0x2]
r4,r5,r6,r7,r8,r9,r10,r11,pc}
```



Function name	Memory address	Heat
adpcm_coder	0x8194	7438904
__flsbuf	0xa314	3724022
fwrite	0xacc0	813580
fseek	0x98d8	488807
__filbuf	0x9fe8	332030
fflush	0x9780	318120
_fflush	0x8568	214537
__rt_memcpy	0xadc8	158849
fread	0xac04	147601
_flushlinebuffered	0x8534	119886
_deferredlazyseek	0x8604	65002
_sys_istty	0x9f2c	55362
_writebuf	0x8468	42560
_sys_read	0x9f0c	36888
_sys_write	0x9eec	12160
main	0x80a8	3426
__main	0x8000	2166
__vfprintf	0x8f70	444
_printf_display	0x8a20	192
freopen	0x871c	159
fclose	0x8668	156

Functionality of Lancet



R0,R4,R8,R11

```
0x81bc :  LDRSH    r12,[r8],#0x2
0x81c0 :  MOV      r10,r11,ASR #3
0x81c4 :  SUB      r12,r12,r0
0x81c8 :  CMP      r12,#0
0x81cc :  MOVGE    r4,#0  exec 77083
0x81d0 :  MOVLT    r4,#0x8  exec 70437
0x81d4
0x81d8
0x81dc
0x81e0
0x81e4
0x81e8
0x81ec
```

Instrumenting before:
0x81bc : LDRSH r12,[r8],#0x2

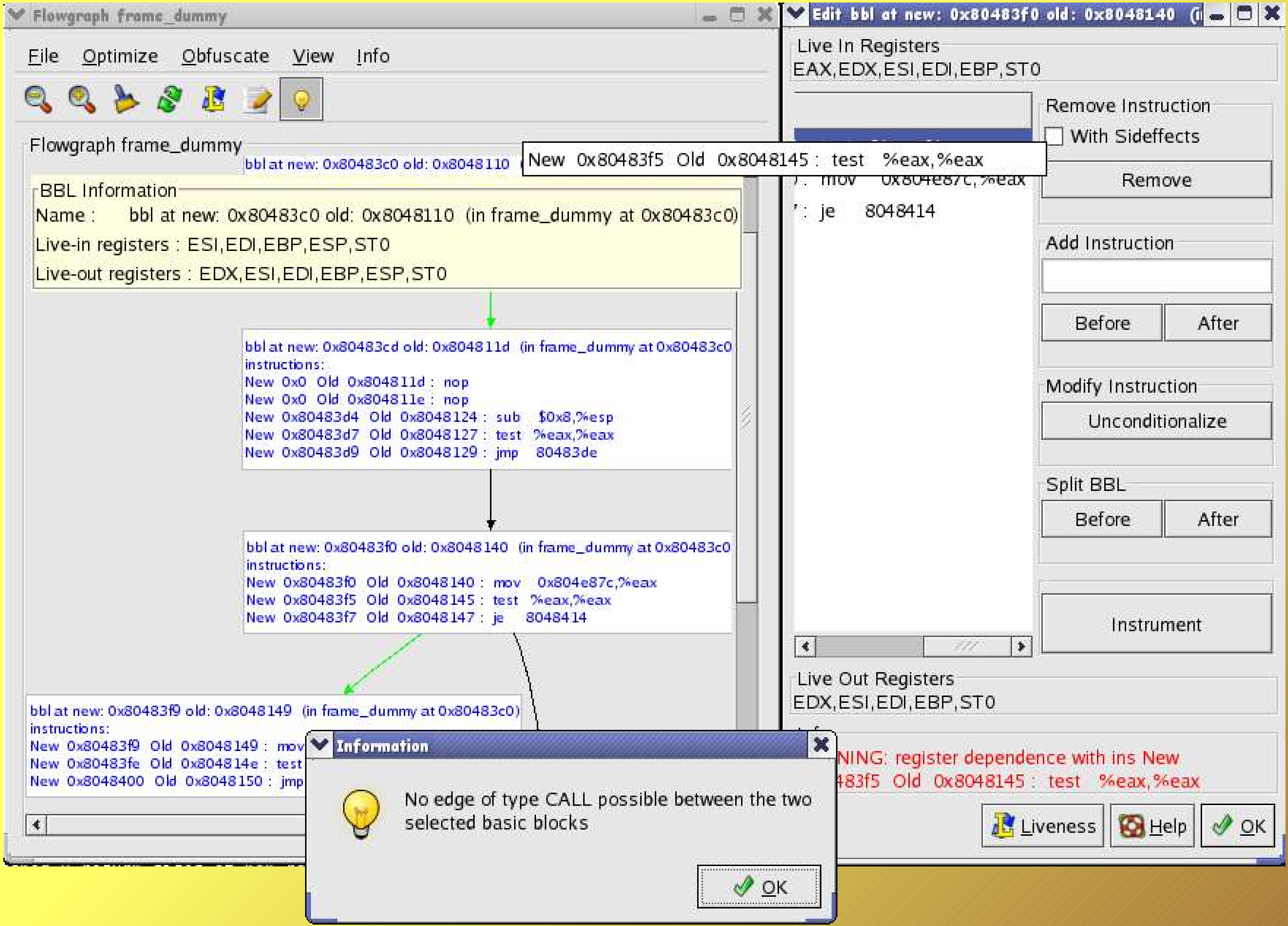
PrintReg

programcounter

REG_R8

empty

Functionality of Lancet



Flowgraph frame_dummy

```
bbl at new: 0x80483c0 old: 0x8048110
```

BBL Information
Name : bbl at new: 0x80483c0 old: 0x8048110 (in frame_dummy at 0x80483c0)
Live-in registers : ESI,EDI,EBP,ESP,ST0
Live-out registers : EDX,ESI,EDI,EBP,ESP,ST0

```
bbl at new: 0x80483cd old: 0x804811d  (in frame_dummy at 0x80483c0
instructions:
New  0x0  Old  0x804811d :  nop
New  0x0  Old  0x804811e :  nop
New  0x80483d4  Old  0x8048124 :  sub   $0x8,%esp
New  0x80483d7  Old  0x8048127 :  test  %eax,%eax
New  0x80483d9  Old  0x8048129 :  jmp   80483de
```

```
bbl at new: 0x80483f0 old: 0x8048140  (in frame_dummy at 0x80483c0
instructions:
New  0x80483f0  Old  0x8048140 :  mov   0x804e87c,%eax
New  0x80483f5  Old  0x8048145 :  test  %eax,%eax
New  0x80483f7  Old  0x8048147 :  je    8048414
```

```
bbl at new: 0x80483f9 old: 0x8048149  (in frame_dummy at 0x80483c0)
instructions:
New  0x80483f9  Old  0x8048149 :  mov
New  0x80483fe  Old  0x804814e :  test
New  0x8048400  Old  0x8048150 :  jmp
```

Live In Registers
EAX,EDX,ESI,EDI,EBP,ST0

```
New  0x80483f5  Old  0x8048145 :  test  %eax,%eax
):  mov   0x804e87c,%eax
':  je    8048414
```

Live Out Registers
EDX,ESI,EDI,EBP,ST0

NING: register dependence with ins New
483f5 Old 0x8048145 : test %eax,%eax

No edge of type CALL possible between the two selected basic blocks

User scenarios

What could you use this functionality for?

→ easy navigation through callgraph, control flow graphs, linker map
→ supervised editing of assembly code of bbl's
→ perform graph operations
→ visual profile feedback
→ graphical instrumentation interface
→ present program analysis information
→ display cfg's before/after/during optimization

User scenarios

Program visualization and analysis
(detect bottlenecks, teaching,...)

→ *easy navigation through callgraph, control flow graphs, linker map*
→ supervised editing of assembly code of bbl's
→ perform graph operations
→ *visual profile feedback*
→ graphical instrumentation interface
→ *present program analysis information*
→ display cfg's before/after/during optimization

User scenarios

Program surgery
(manual program edits supported by feedback)

→ *easy navigation through callgraph, control flow graphs, linker map*
→ *supervised editing of assembly code of bbl's*
→ *perform graph operations*
→ *visual profile feedback*
→ graphical instrumentation interface
→ *present program analysis information*
→ display cfg's before/after/during optimization

User scenarios

Assist in development of new transformations
(compiler research, ...)

→ *easy navigation through callgraph, control flow graphs, linker map*
→ *supervised editing of assembly code of bbl's*
→ *perform graph operations*
→ visual profile feedback
→ graphical instrumentation interface
→ *present program analysis information*
→ *display cfg's before/after/during optimization*

User scenarios

Visualization and steering of transformations
(interactive program optimization,...)

→ easy navigation through callgraph, control flow graphs, linker map
→ supervised editing of assembly code of bbl's
→ perform graph operations
→ visual profile feedback
→ graphical instrumentation interface
→ *present program analysis information*
→ *display cfg's before/after/during optimization*
(future work: adapt transformations to enable interaction)

User scenarios

Fine-grained, point-wise instrumentation
(analysis, debugging, profiling, ...)

- → *easy navigation through callgraph, control flow graphs, linker map*
- → supervised editing of assembly code of bbl's
- → perform graph operations
- → visual profile feedback
- → *graphical instrumentation interface*
- → present program analysis information
- → display cfg's before/after/during optimization

Future work

- present additional program analysis information
- provide 'undo' functionality
- integrate instrumentation & transformation feedback
- provide an interface for optimization reordering
- ...

Questions?

http://www.elis.ugent.be/diablo