Design of Embedded Hardware and Firmware

Software Optimizations

Andrea Guerrieri

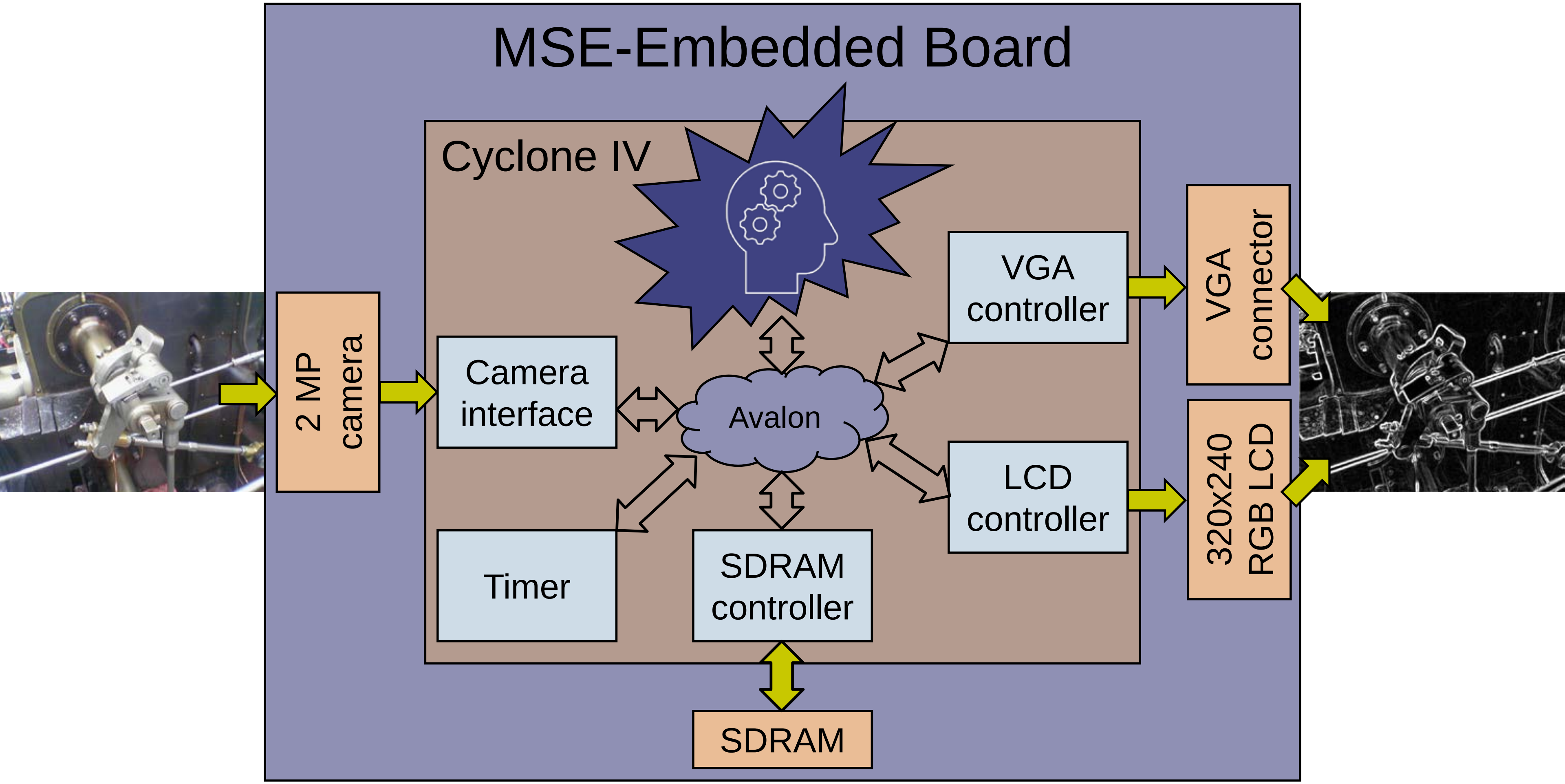
HES-SO//Genève

andrea.guerrieri@hesge.ch

Problem specifications

We need real-time edge-detection on a live video stream

Problem specifications



Initial implementation

Function	Cycles/image	Cycles/pixel
conv_grayscale	127336549	~648
sobel_x	729624798	~3711
sobel_y	729677739	~3711
sobel_threshold	102849150	~523
TOTAL	**1689488236**	**8593**

Camera:

- **RGB565**
- **512x384**
- **5-15 fps**

⇨

vs.

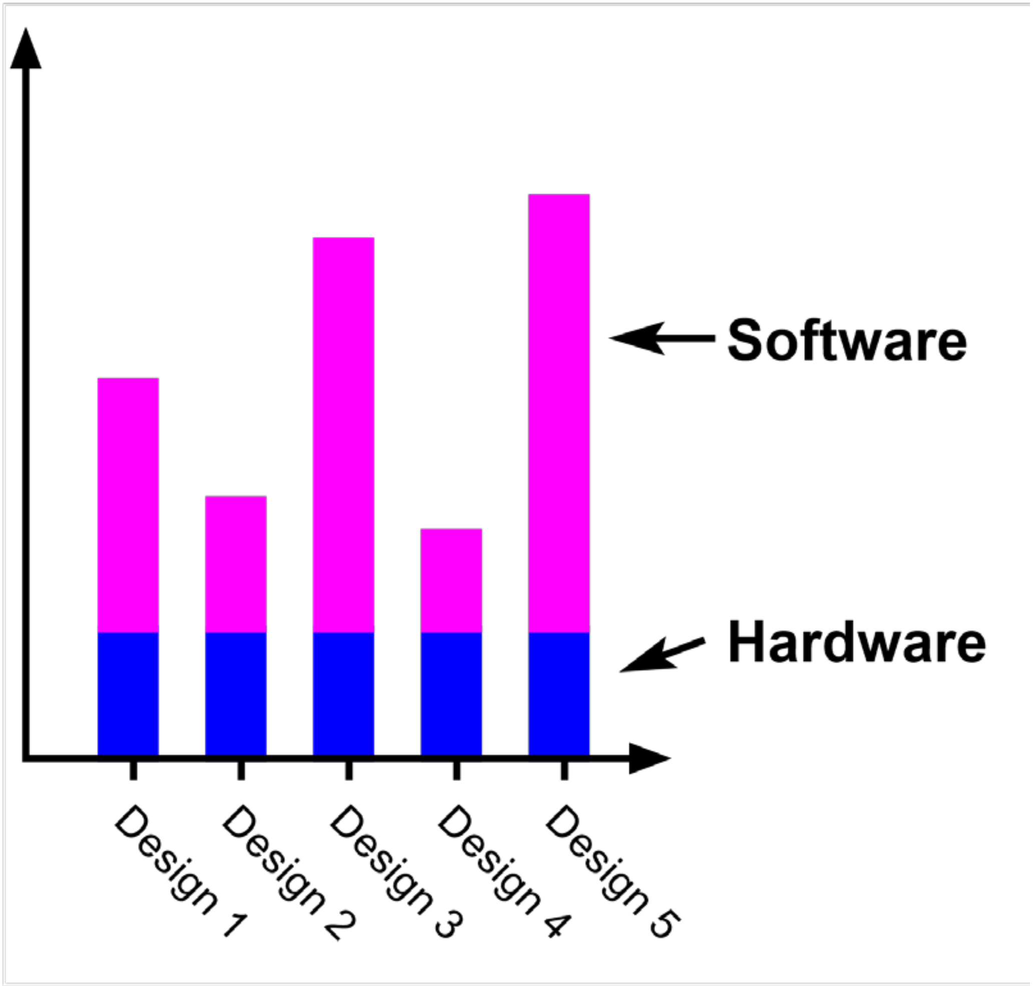
17 cycles/pixel

(~505x improvement)

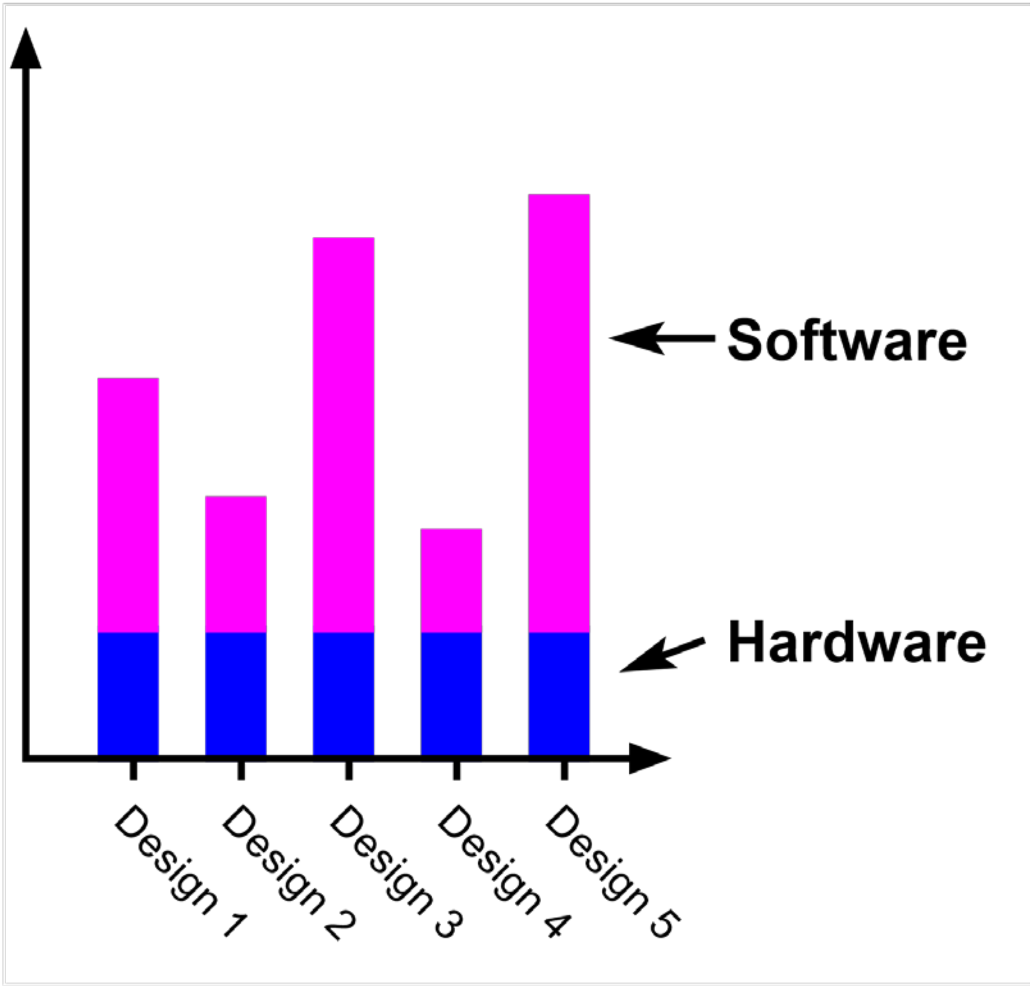
Naive approach

Software Optimizations

Traditional approach



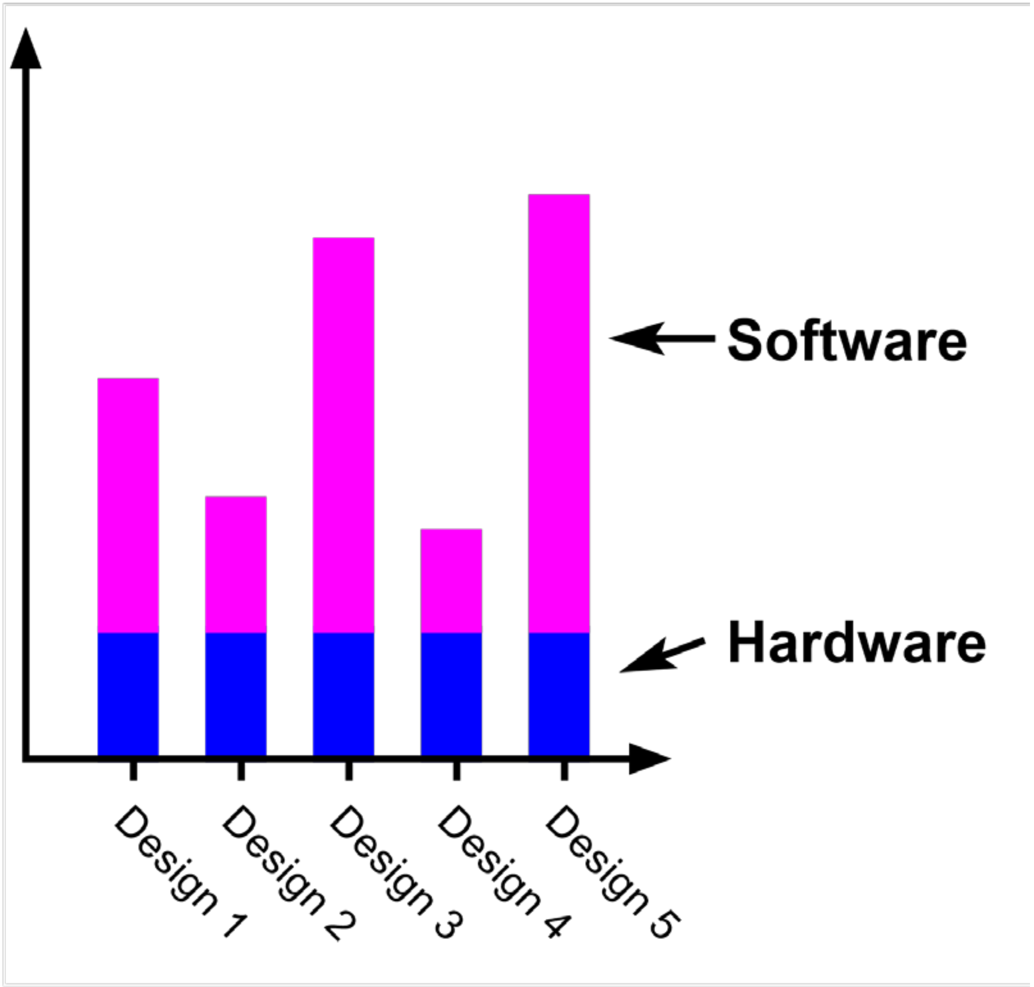
Traditional approach



SW optimization

- ➢ What do we need to optimize?

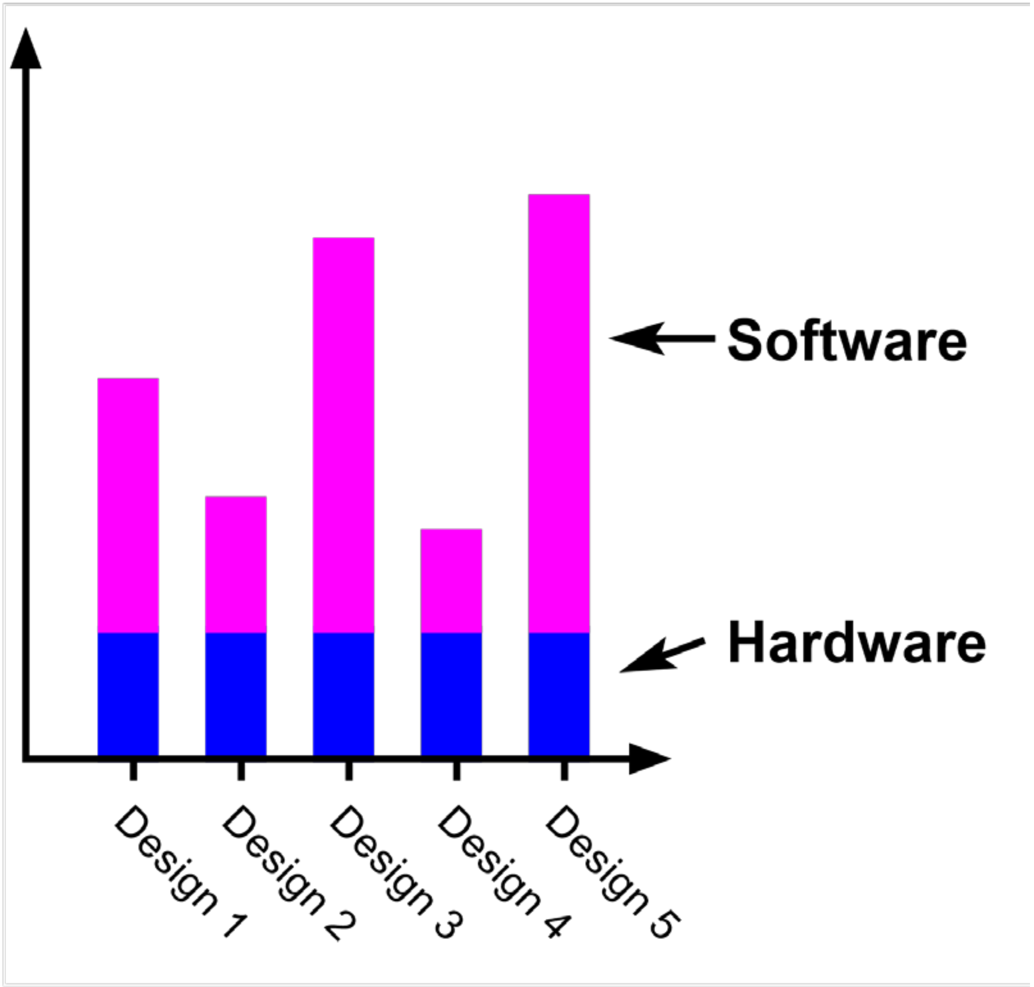
Traditional approach



SW optimization

- What do we need to optimize?
- Do we need to understand completely the algorithm?

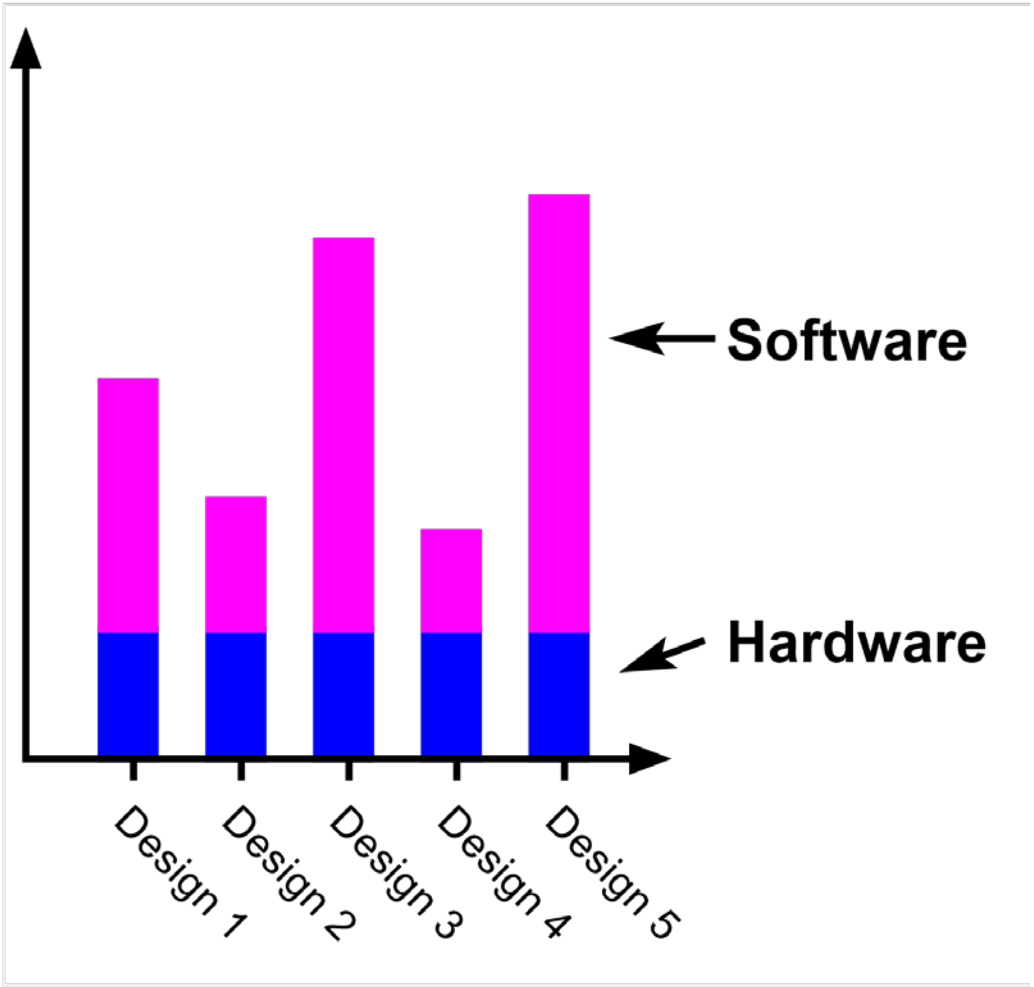
Traditional approach



SW optimization

- What do we need to optimize?
- Do we need to understand completely the algorithm?
- Which tools can we use?

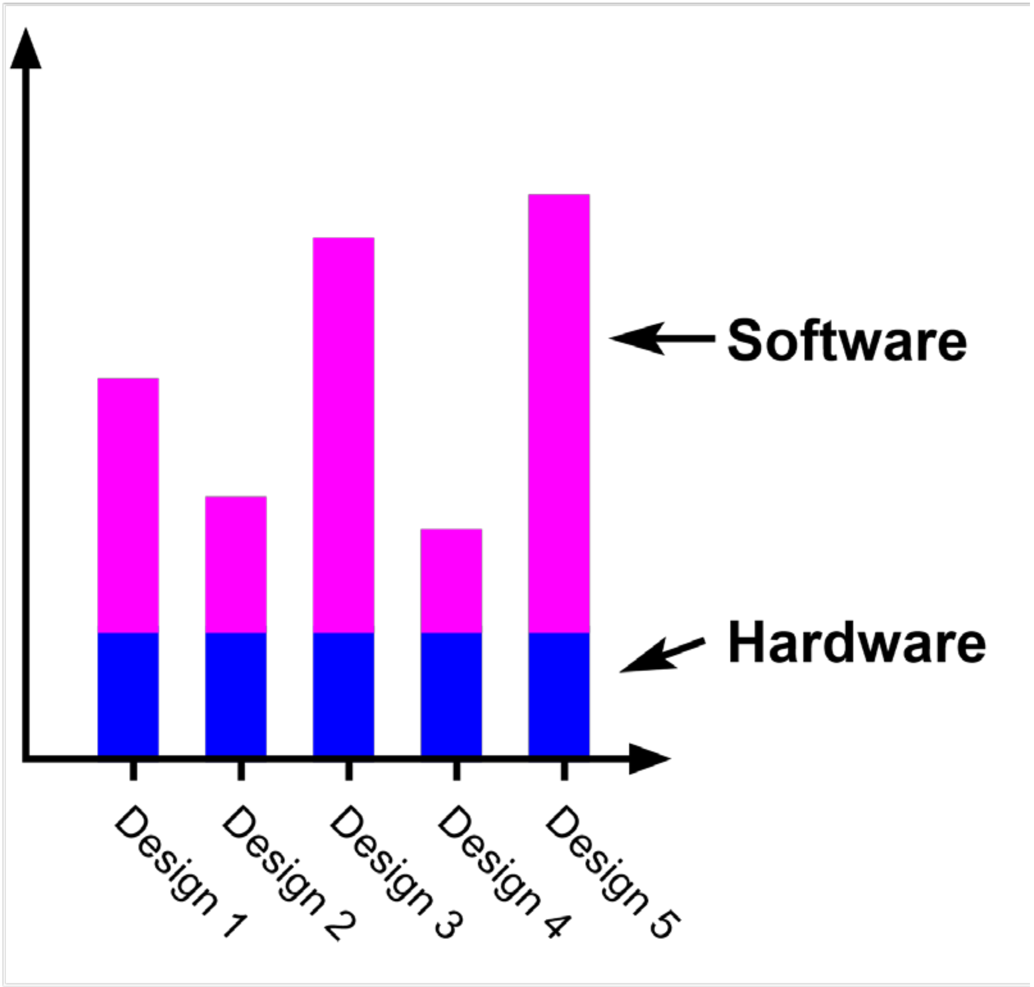
Traditional approach



SW optimization

- What do we need to optimize?
- Do we need to understand completely the algorithm?
- Which tools can we use?
- Where do we focus?

Traditional approach



SW optimization

- What do we need to optimize?
- Do we need to understand completely the algorithm?
- Which tools can we use?
- Where do we focus?
- Which details are more important?

Naive approach

Compiler optimizations (Out-of-the-box)

Compilation Flags (CFLAGS)

- Generic

  -O0 -O1 -O3 –Ofast –Omax –Omin…

- Architecture Specific

  -mthumb –march -mfloatabi…

- Debugging options

  -g…

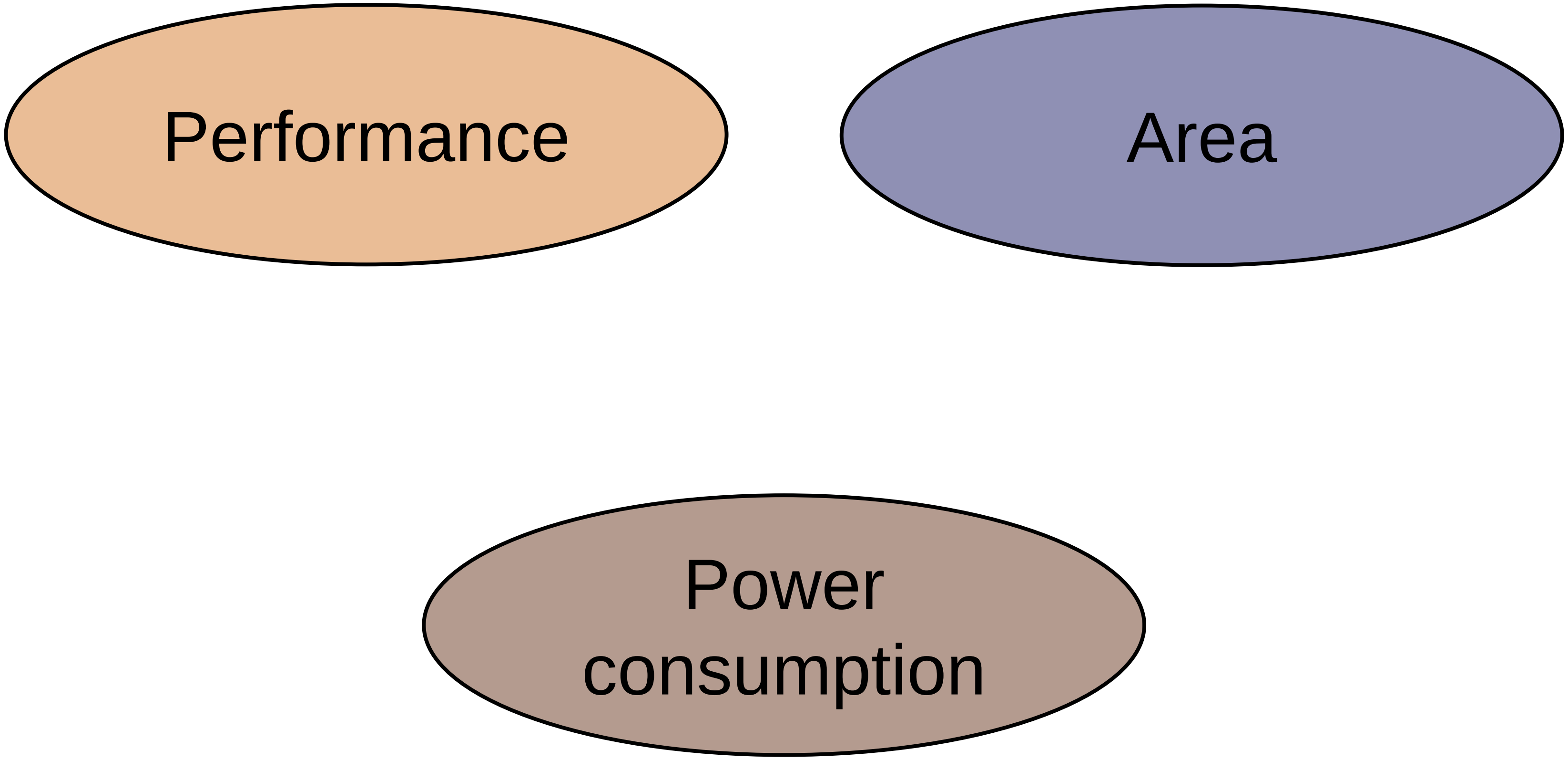
«Out-of-the-box» optimization

Function	-O0	-O1	-O2	-O3
conv_grayscale (Time)	127336549 (2.547 s)	44161114 (0.883 s)	41296763 (0.826 s)	41194172 (0.824 s)
sobel_x (Time)	729624798 (14.592 s)	129966322 (2.599 s)	109616923 (2.192 s)	29727206 (0.595 s)
sobel_y (Time)	729677739 (14.594 s)	129964177 (2.599 s)	109607580 (2.192 s)	27249940 (0.545 s)
sobel_threshold (Time)	102849150 (2.057 s)	33664258 (0.673 s)	32015554 (0.64 s)	32126201 (0.643 s)
TOTAL (Time)	**1689488236 (33.79 s)**	**337755871 (6.754 s)**	**292536820 (5.85 s)**	**130297519 (2.607 s)**

«Out-of-the-box» optimization

Function	-O0	-O1	-O2	-O3	
conv_grayscale (Time)	127336549 (2.547 s)	44161114 (0.883 s)	41296763 (0.826 s)	41194172 (0.824 s)	3.09x
sobel_x (Time)	729624798 (14.592 s)	129966322 (2.599 s)	109616923 (2.192 s)	29727206 (0.595 s)	24.54x
sobel_y (Time)	729677739 (14.594 s)	129964177 (2.599 s)	109607580 (2.192 s)	27249940 (0.545 s)	26.78x
sobel_threshold (Time)	102849150 (2.057 s)	33664258 (0.673 s)	32015554 (0.64 s)	32126201 (0.643 s)	3.20x
TOTAL (Time)	**1689488236 (33.79 s)**	**337755871 (6.754 s)**	**292536820 (5.85 s)**	**130297519 (2.607 s)**	12.97x

Solution

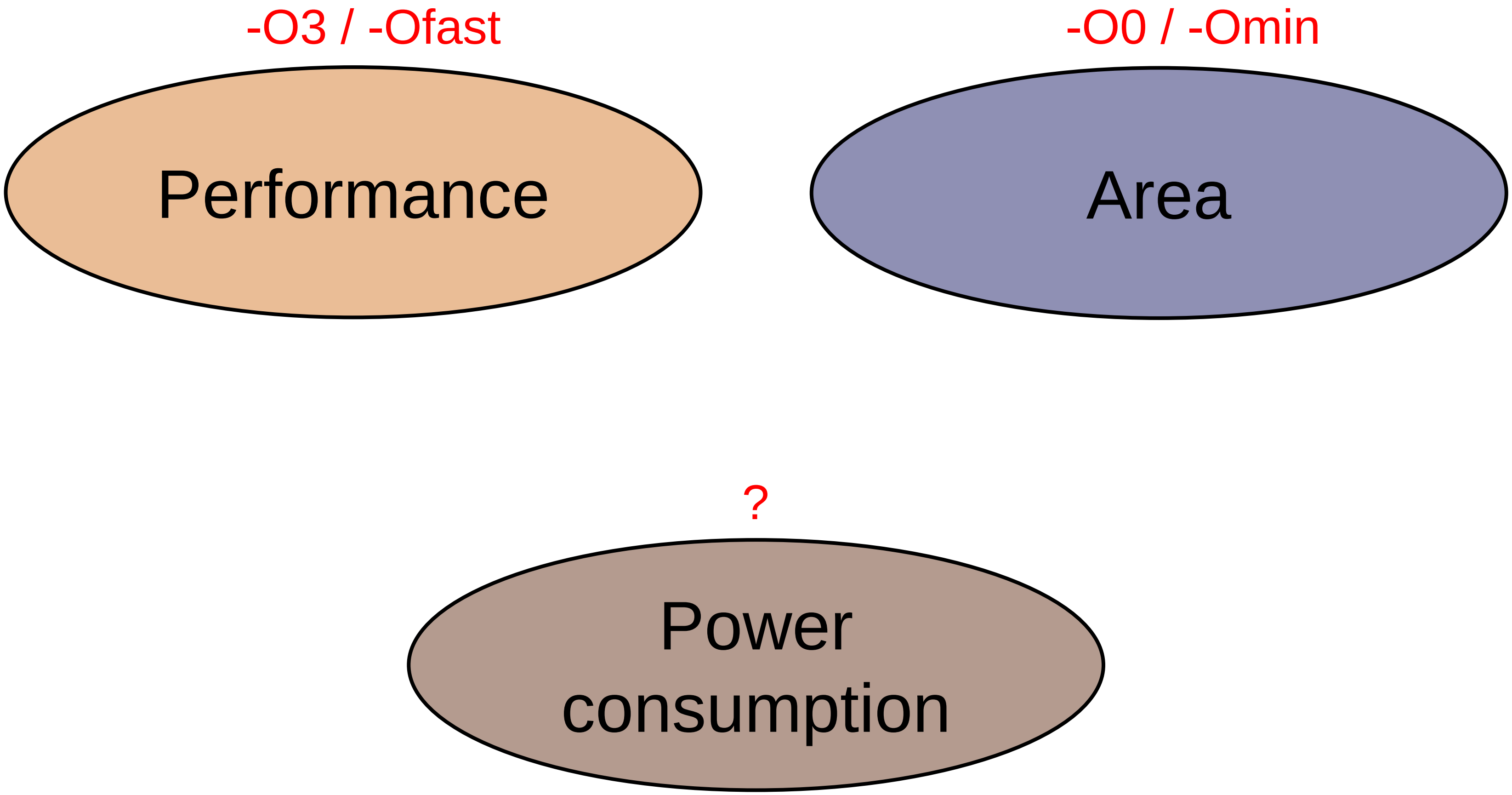


Performance

Area

Power consumption

Solution



-O3 / -Ofast

Performance

-O0 / -Omin

Area

?

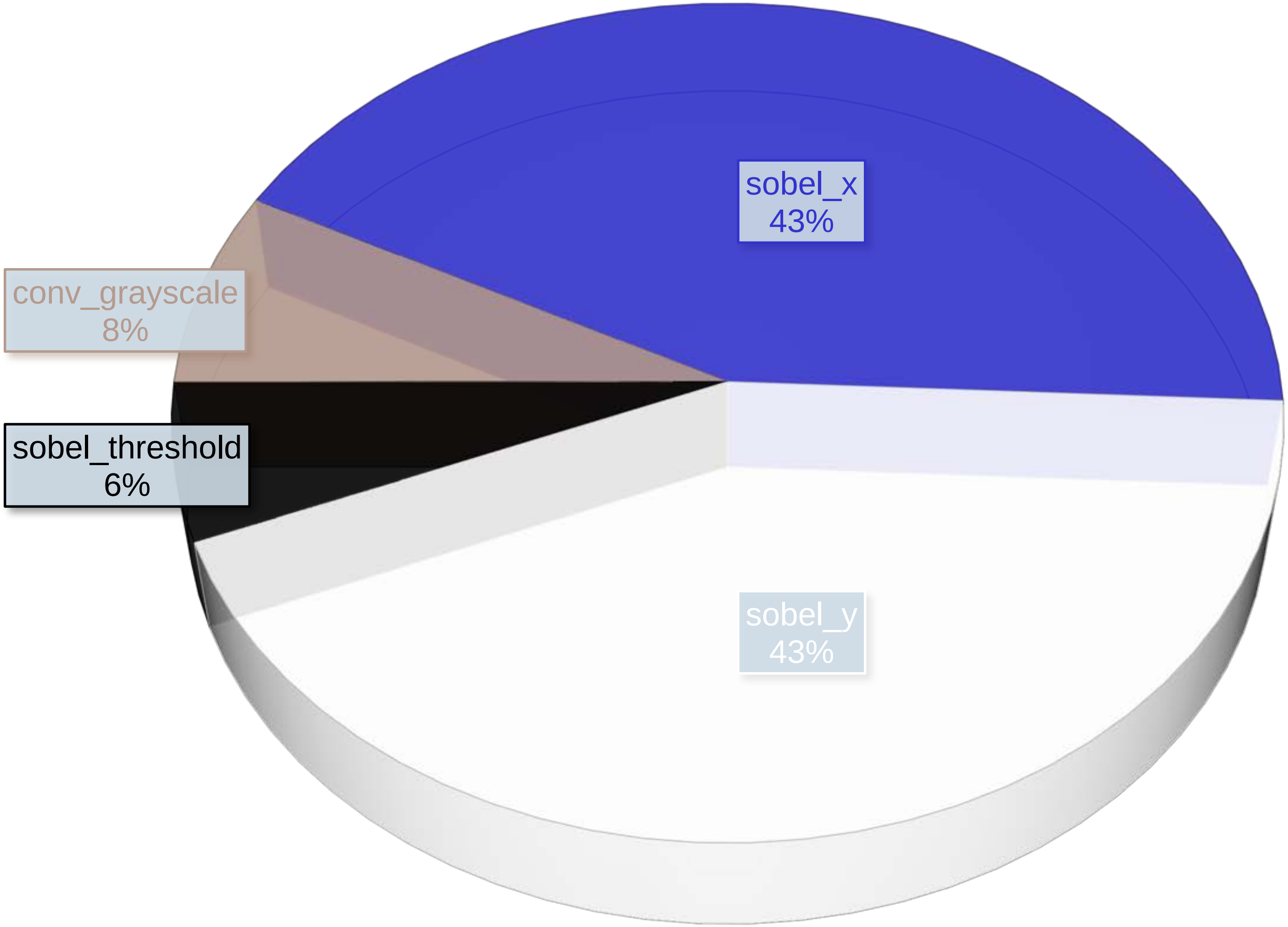
Power consumption

Naive approach

Is this enough?

Better to have a look inside…

What do we need to optimize?



Which tools can we use?

- How do we get the data for the previous chart?

Which tools can we use?

- How do we get the data for the previous chart?
- We could count instructions (even automatically in the compiler)

Which tools can we use?

- How do we get the data for the previous chart?
- We could count instructions (even automatically in the compiler)
- However:
  - Data dependent loops or jumps
  - Pointers and dynamic memory management

Which tools can we use?

- How do we get the data for the previous chart?
- We could count instructions (even automatically in the compiler)
- However:
  - Data dependent loops or jumps
  - Pointers and dynamic memory management

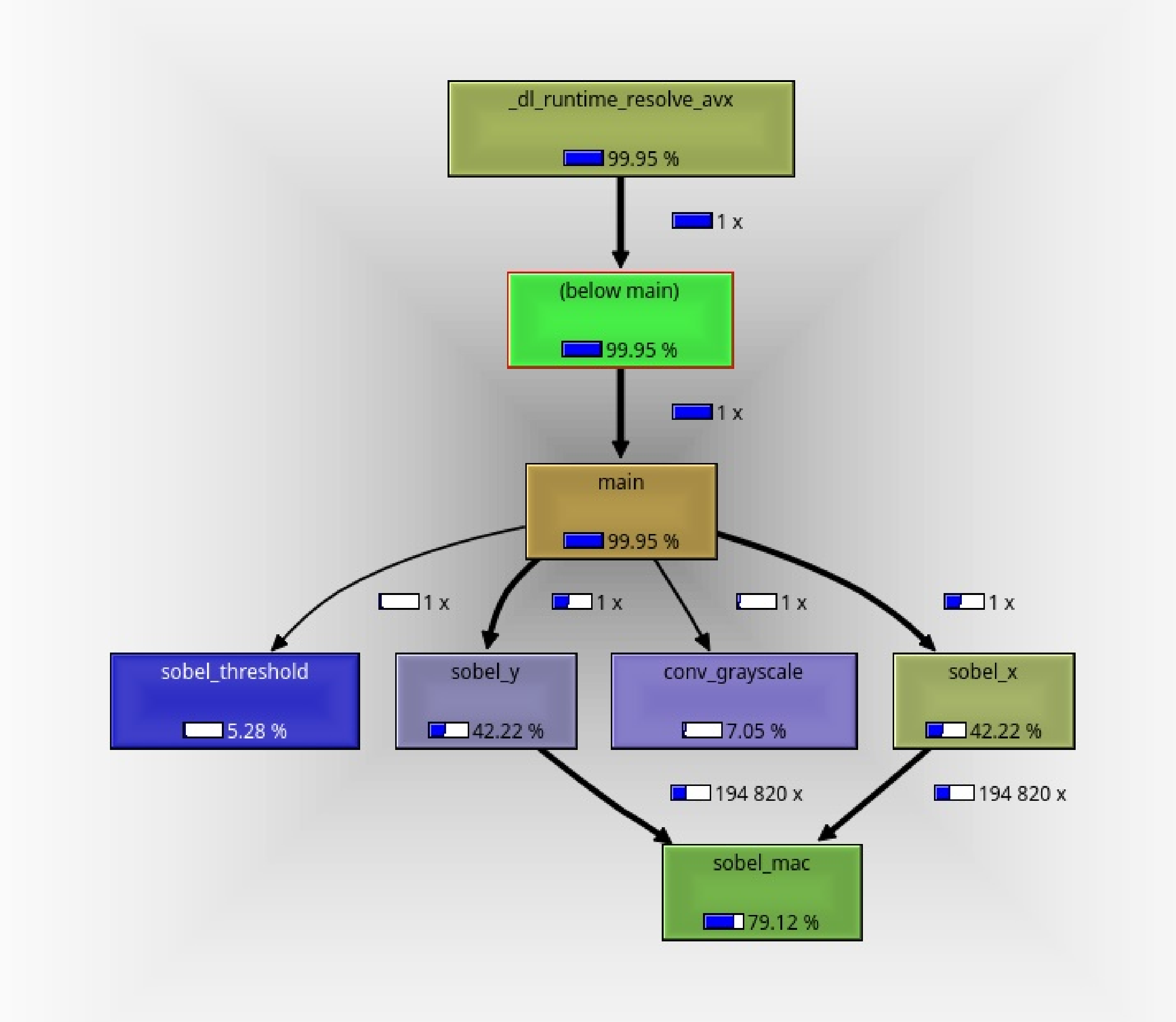
- We need to check the dynamic behaviour

Which tools can we use?

- How do we get the data for the previous chart?
- We could count instructions (even automatically in the compiler)
- However:
  - Data dependent loops or jumps
  - Pointers and dynamic memory management

- We need to check the dynamic behaviour
- Inserting “counters” in the code for each section of interest: ***Profiling***
- Tools like: gprof, valgrind, k(q)cachegrind…

Profiling with kcachegrind (on PC)



Profiling (on PC)

- Relatively easy to set-up and good results

Profiling (on PC)

- Relatively easy to set-up and good results

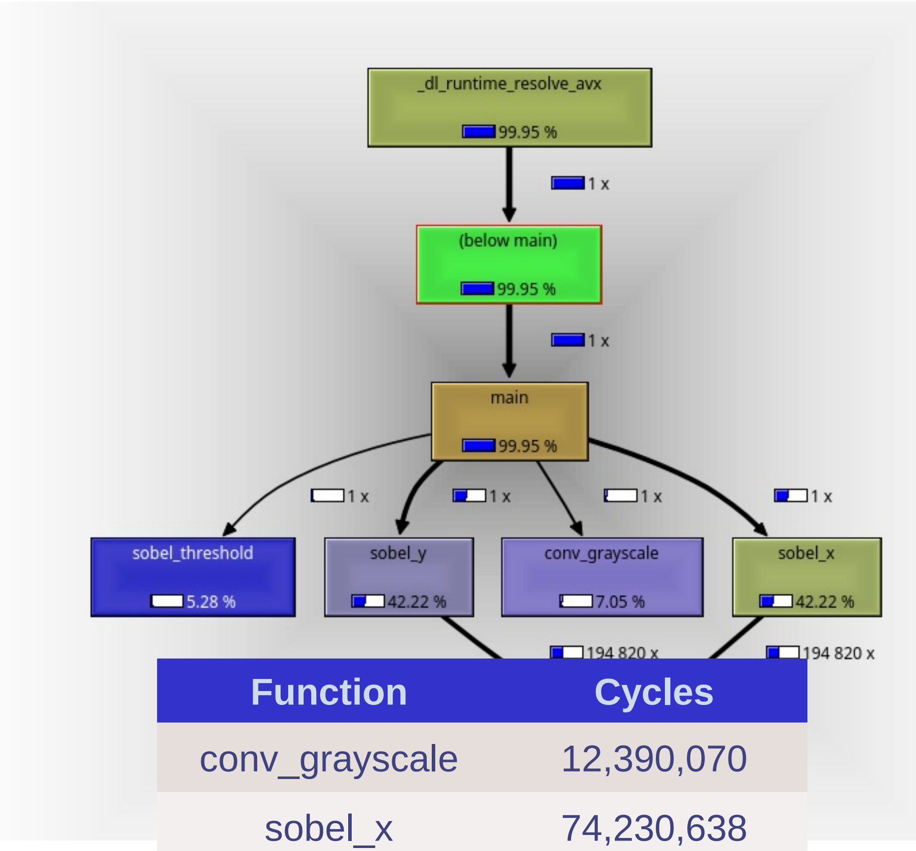
- However:
  - Results specific for the used data set. Representative?

Profiling (on PC)

- Relatively easy to set-up and good results

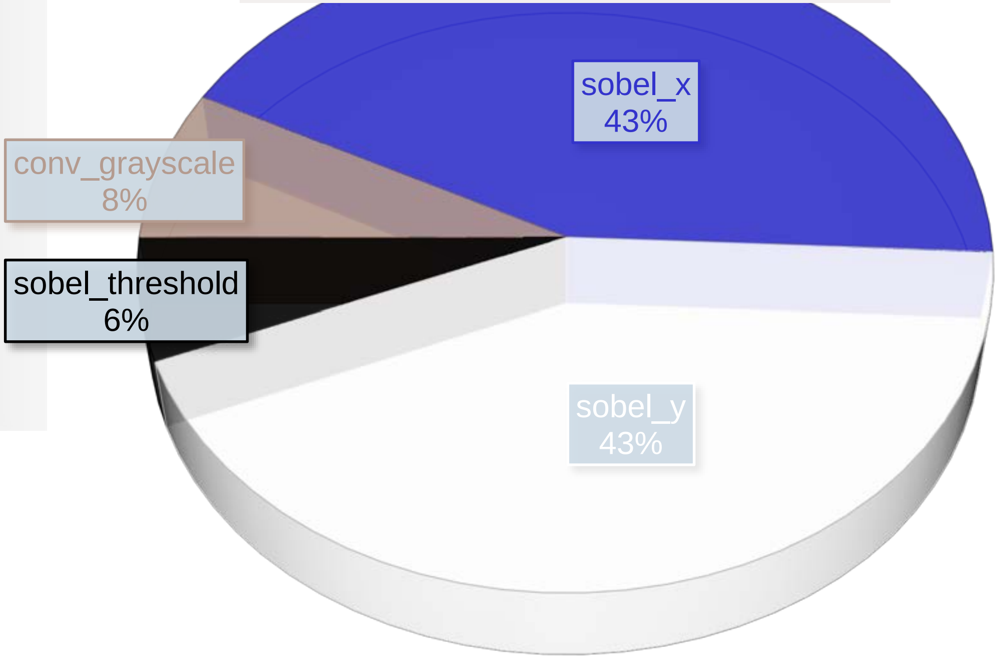
- However:
  - Results specific for the used data set. Representative?
  - Different compiler, architecture. Will it be the same in the HW?

Profiling (on PC)

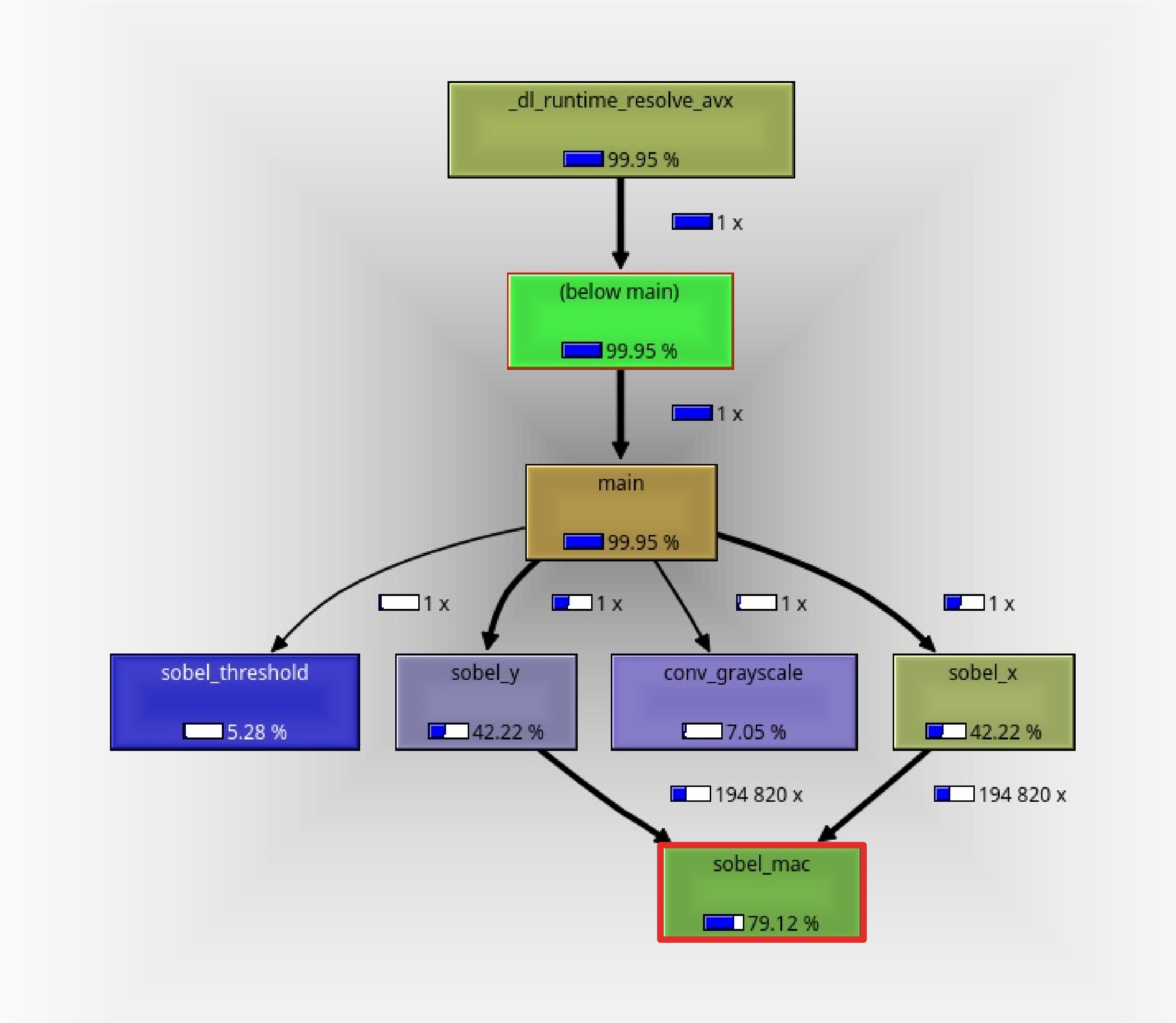


Function	Cycles
conv_grayscale	12,390,070
sobel_x	74,230,638
sobel_y	74,230,638
sobel_threshold	9,286,527
TOTAL	**170,137,873**

Function	Cycles
conv_grayscale	127,336,549
sobel_x	729,624,798
sobel_y	729,677,739
sobel_threshold	102,849,150
TOTAL	**1,689,488,236**

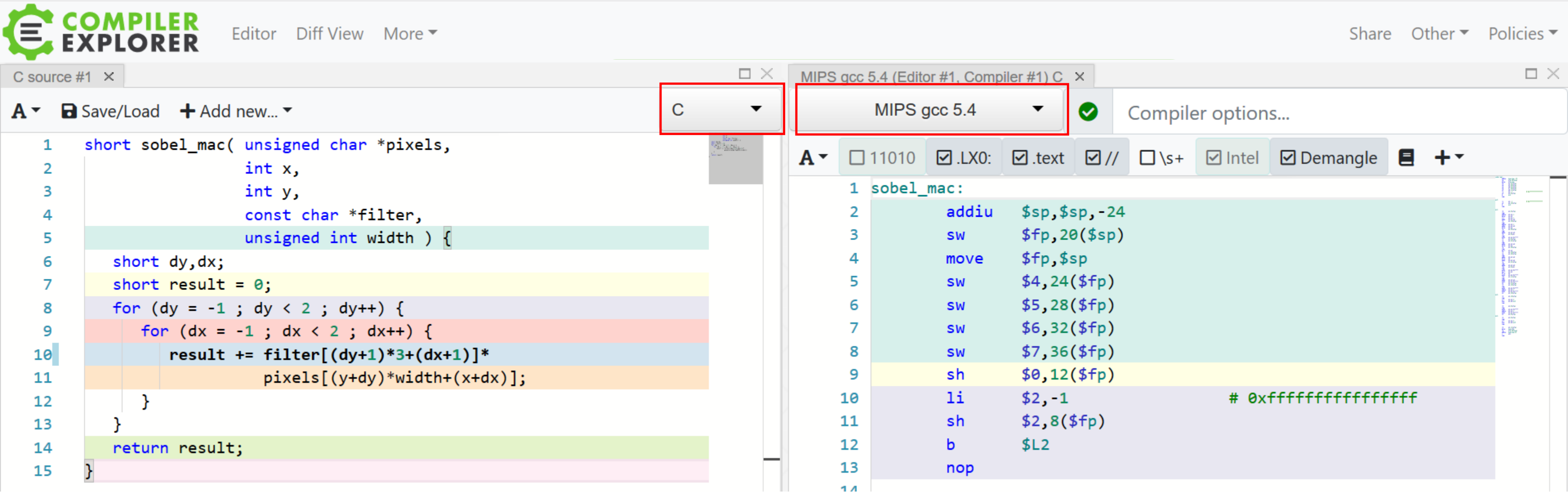


Profiling with kcachegrind (on PC)



Which tools can we use?

- Try to understand the generated assembly by the compiler
- Depends on the processor architecture and compiler, but we can have a quick look to what it looks like…



C

MIPS gcc 5.4

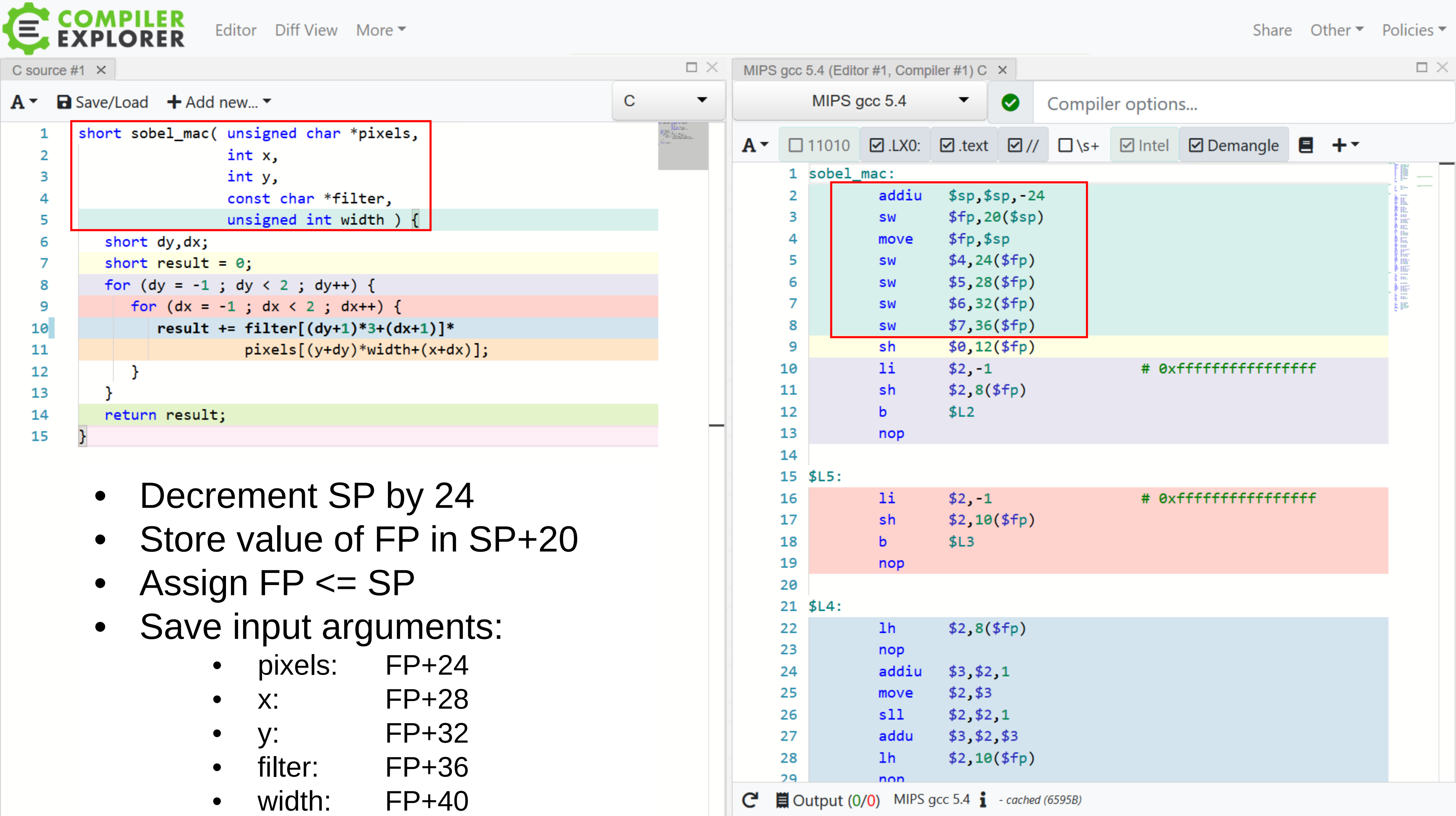
Compiler options...

□ 11010 ☑ .LX0: ☑ .text ☑ // □ \s+ ☑ Intel ☑ Demangle

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
sobel_mac:
        addiu   $sp,$sp,-24
        sw      $fp,20($sp)
        move    $fp,$sp
        sw      $4,24($fp)
        sw      $5,28($fp)
        sw      $6,32($fp)
        sw      $7,36($fp)
        sh      $0,12($fp)
        li      $2,-1                   # 0xffffffffffffffff
        sh      $2,8($fp)
        b       $L2
        nop
```

Compiler explorer

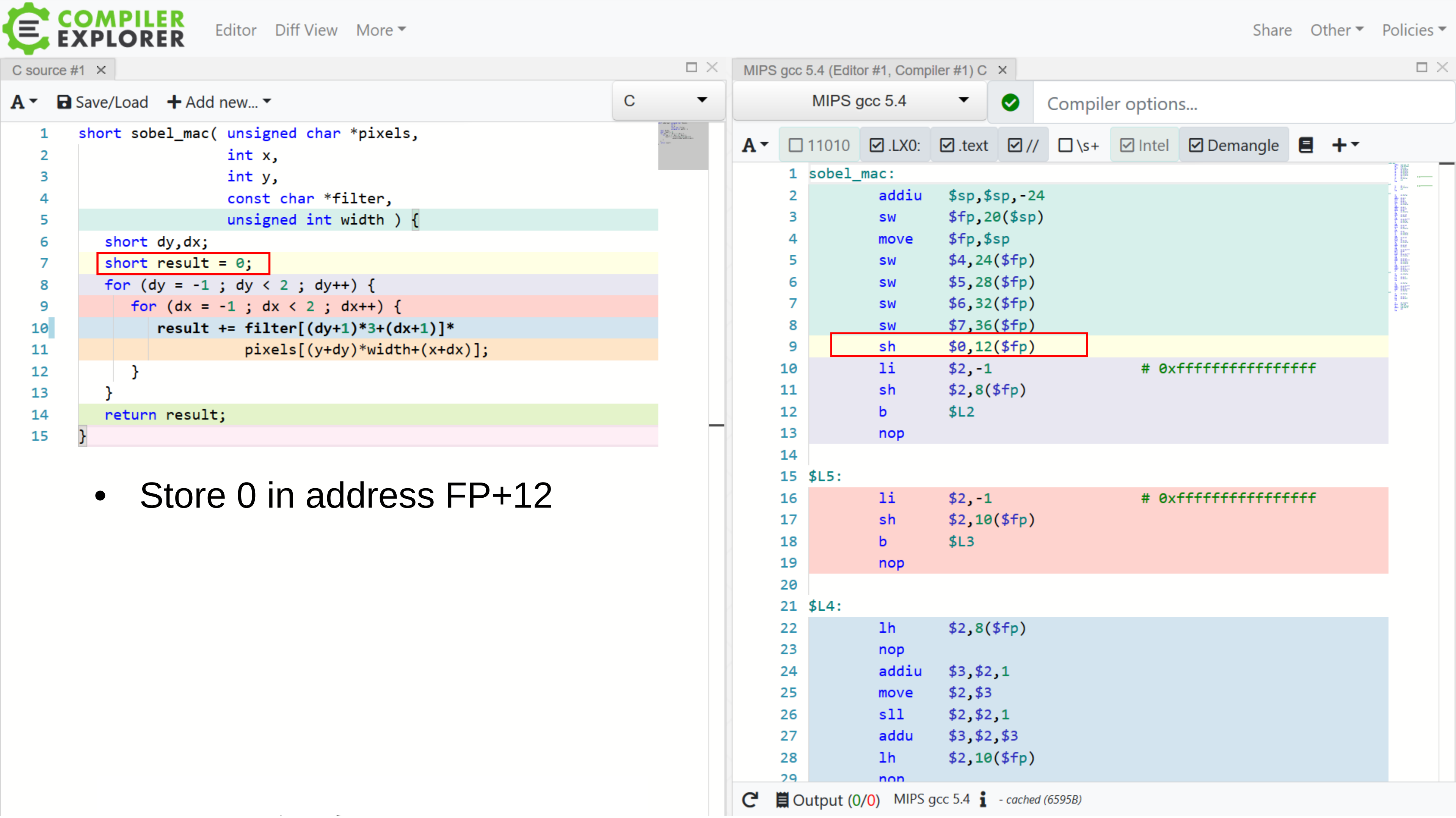


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
sobel_mac:
        addiu   $sp,$sp,-24
        sw      $fp,20($sp)
        move    $fp,$sp
        sw      $4,24($fp)
        sw      $5,28($fp)
        sw      $6,32($fp)
        sw      $7,36($fp)
        sh      $0,12($fp)
        li      $2,-1                   # 0xffffffffffffffff
        sh      $2,8($fp)
        b       $L2
        nop

$L5:
        li      $2,-1                   # 0xffffffffffffffff
        sh      $2,10($fp)
        b       $L3
        nop

$L4:
        lh      $2,8($fp)
        nop
        addiu   $3,$2,1
        move    $2,$3
        sll     $2,$2,1
        addu    $3,$2,$3
        lh      $2,10($fp)
        nop
```

- Decrement SP by 24
- Store value of FP in SP+20
- Assign FP <= SP
- Save input arguments:
  - pixels: FP+24
  - x: FP+28
  - y: FP+32
  - filter: FP+36
  - width: FP+40

Compiler explorer



```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

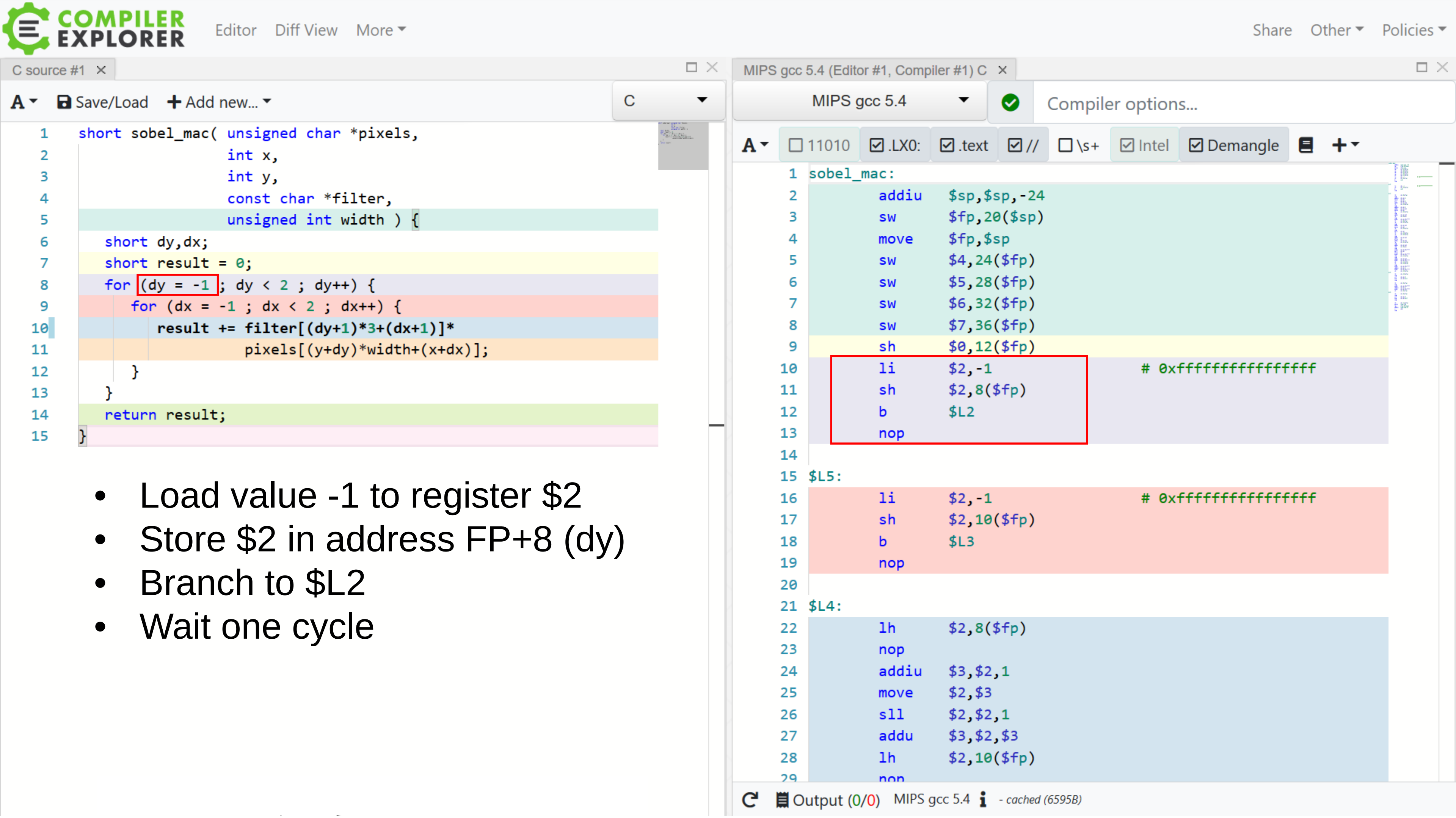
□ 11010 ☑ .LX0: ☑ .text ☑ // □ \s+ ☑ Intel ☑ Demangle

```
sobel_mac:
        addiu   $sp,$sp,-24
        sw      $fp,20($sp)
        move    $fp,$sp
        sw      $4,24($fp)
        sw      $5,28($fp)
        sw      $6,32($fp)
        sw      $7,36($fp)
        sh      $0,12($fp)
        li      $2,-1                  # 0xffffffffffffffff
        sh      $2,8($fp)
        b       $L2
        nop

$L5:
        li      $2,-1                  # 0xffffffffffffffff
        sh      $2,10($fp)
        b       $L3
        nop

$L4:
        lh      $2,8($fp)
        nop
        addiu   $3,$2,1
        move    $2,$3
        sll     $2,$2,1
        addu    $3,$2,$3
        lh      $2,10($fp)
        nop
```

Output (0/0) MIPS gcc 5.4 - cached (6595B)

- Store 0 in address FP+12

Compiler explorer

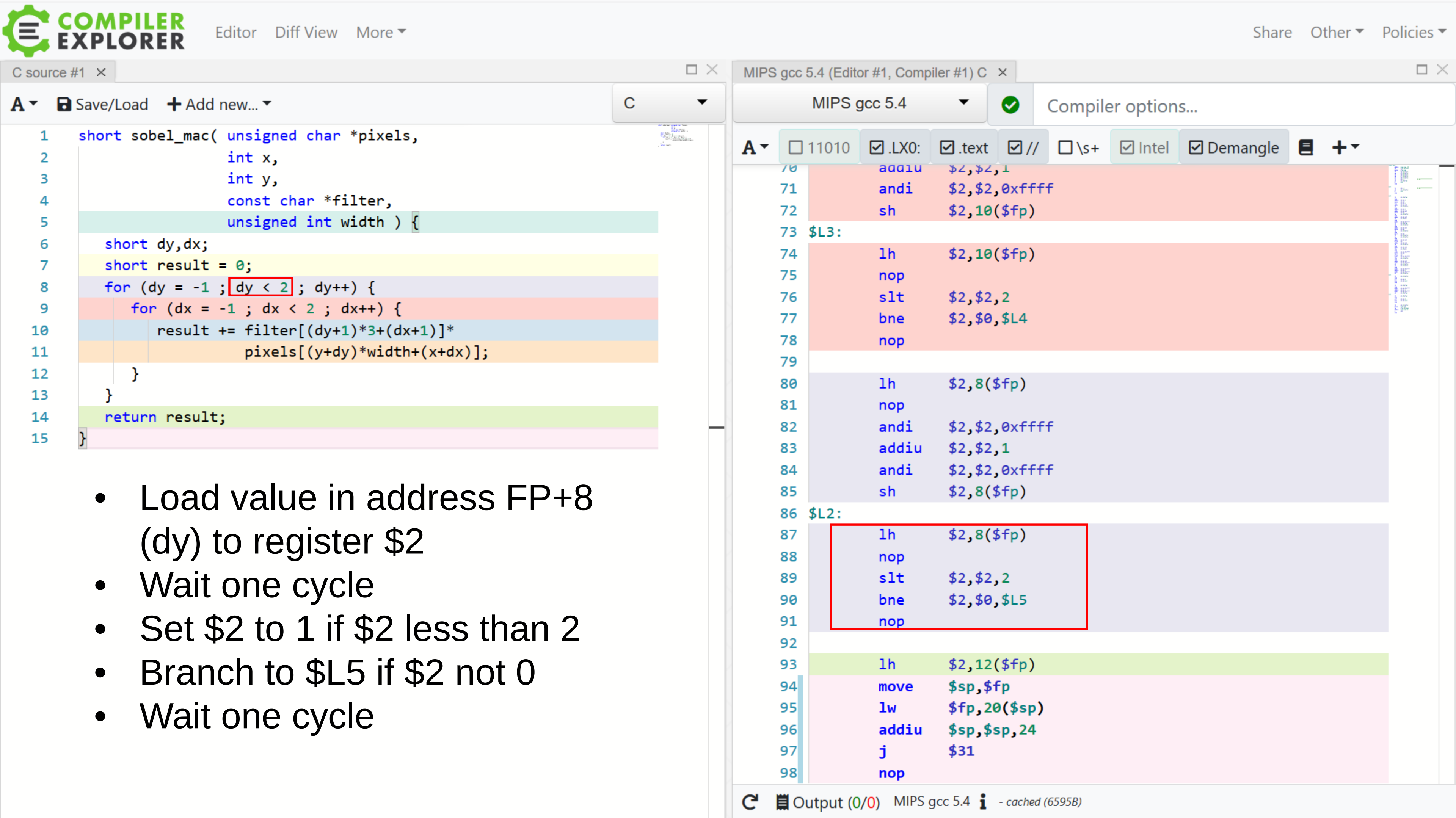


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
sobel_mac:
        addiu   $sp,$sp,-24
        sw      $fp,20($sp)
        move    $fp,$sp
        sw      $4,24($fp)
        sw      $5,28($fp)
        sw      $6,32($fp)
        sw      $7,36($fp)
        sh      $0,12($fp)
        li      $2,-1                   # 0xffffffffffffffff
        sh      $2,8($fp)
        b       $L2
        nop

$L5:
        li      $2,-1                   # 0xffffffffffffffff
        sh      $2,10($fp)
        b       $L3
        nop

$L4:
        lh      $2,8($fp)
        nop
        addiu   $3,$2,1
        move    $2,$3
        sll     $2,$2,1
        addu    $3,$2,$3
        lh      $2,10($fp)
        nop
```

- Load value -1 to register $2
- Store $2 in address FP+8 (dy)
- Branch to $L2
- Wait one cycle

Compiler explorer



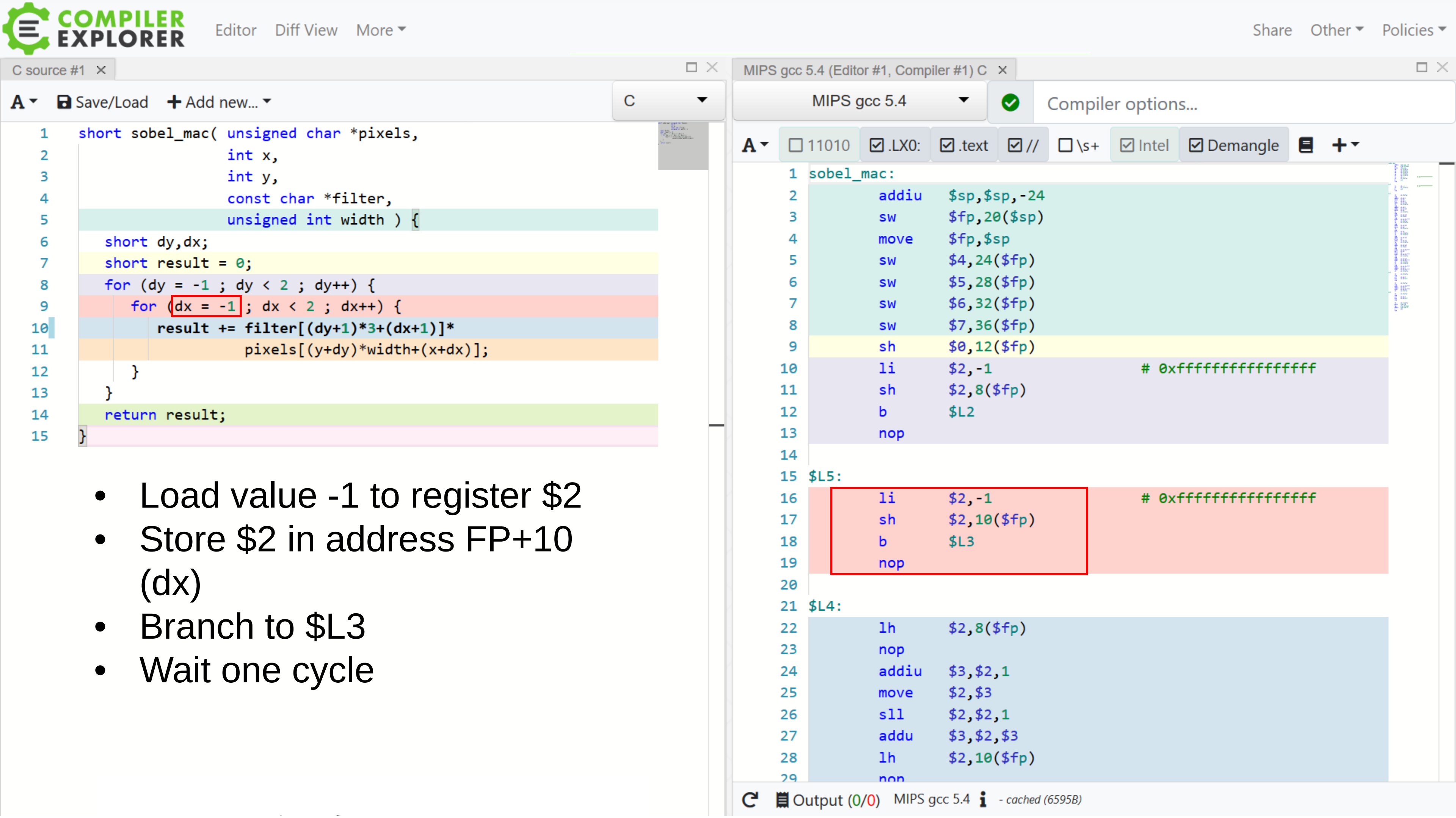
☐ 11010 ☑ .LX0: ☑ .text ☑ // ☐ \s+ ☑ Intel ☑ Demangle

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```asm
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,10($fp)
$L3:
        lh      $2,10($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L4
        nop

        lh      $2,8($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,8($fp)
$L2:
        lh      $2,8($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L5
        nop

        lh      $2,12($fp)
        move    $sp,$fp
        lw      $fp,20($sp)
        addiu   $sp,$sp,24
        j       $31
        nop
```

- Load value in address FP+8 (dy) to register $2
- Wait one cycle
- Set $2 to 1 if $2 less than 2
- Branch to $L5 if $2 not 0
- Wait one cycle

Compiler explorer



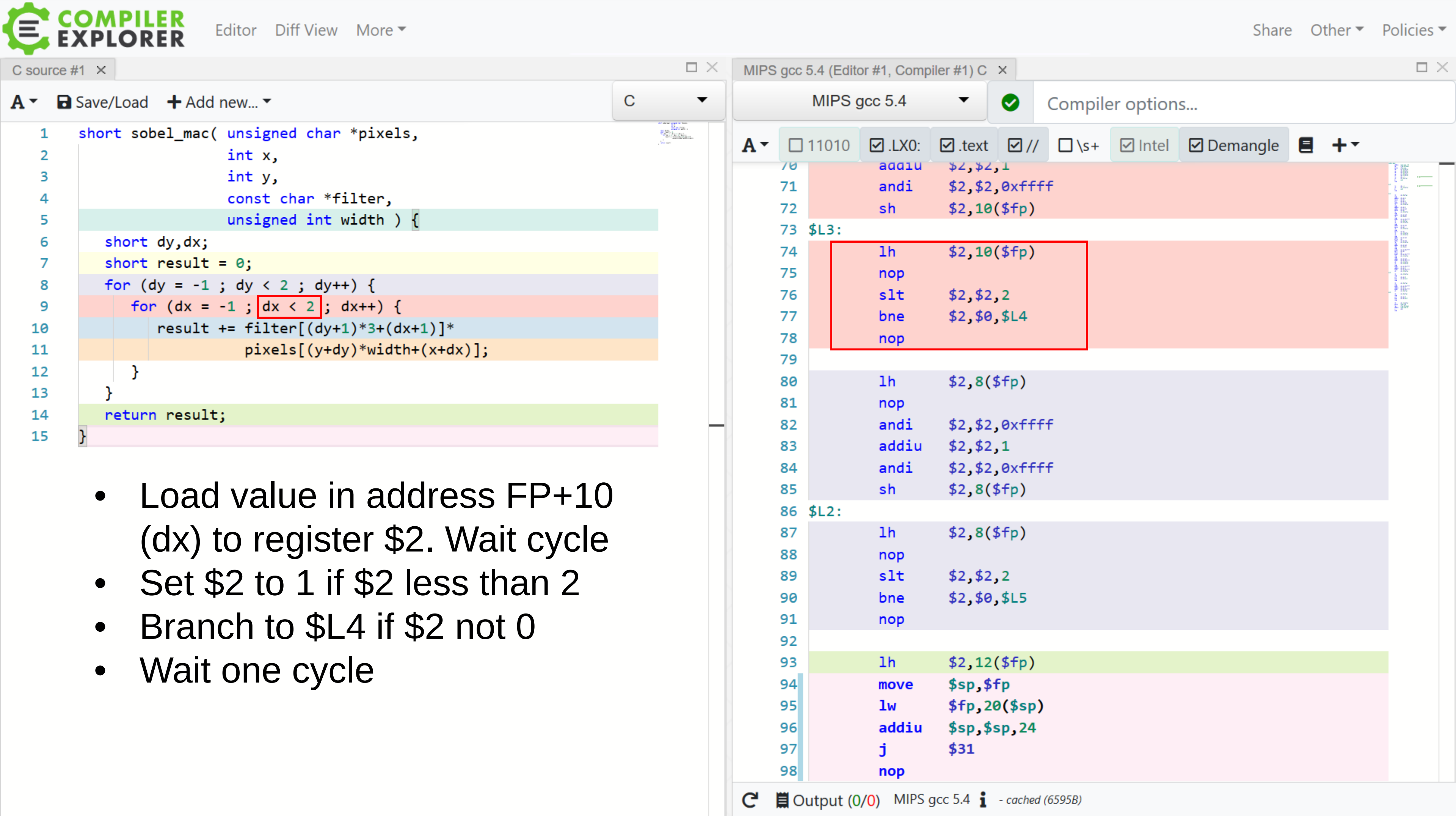
□ 11010 ☑ .LX0: ☑ .text ☑ // □ \s+ ☑ Intel ☑ Demangle

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
sobel_mac:
        addiu   $sp,$sp,-24
        sw      $fp,20($sp)
        move    $fp,$sp
        sw      $4,24($fp)
        sw      $5,28($fp)
        sw      $6,32($fp)
        sw      $7,36($fp)
        sh      $0,12($fp)
        li      $2,-1                   # 0xffffffffffffffff
        sh      $2,8($fp)
        b       $L2
        nop

$L5:
        li      $2,-1                   # 0xffffffffffffffff
        sh      $2,10($fp)
        b       $L3
        nop

$L4:
        lh      $2,8($fp)
        nop
        addiu   $3,$2,1
        move    $2,$3
        sll     $2,$2,1
        addu    $3,$2,$3
        lh      $2,10($fp)
        nop
```

- Load value -1 to register $2
- Store $2 in address FP+10 (dx)
- Branch to $L3
- Wait one cycle

Compiler explorer



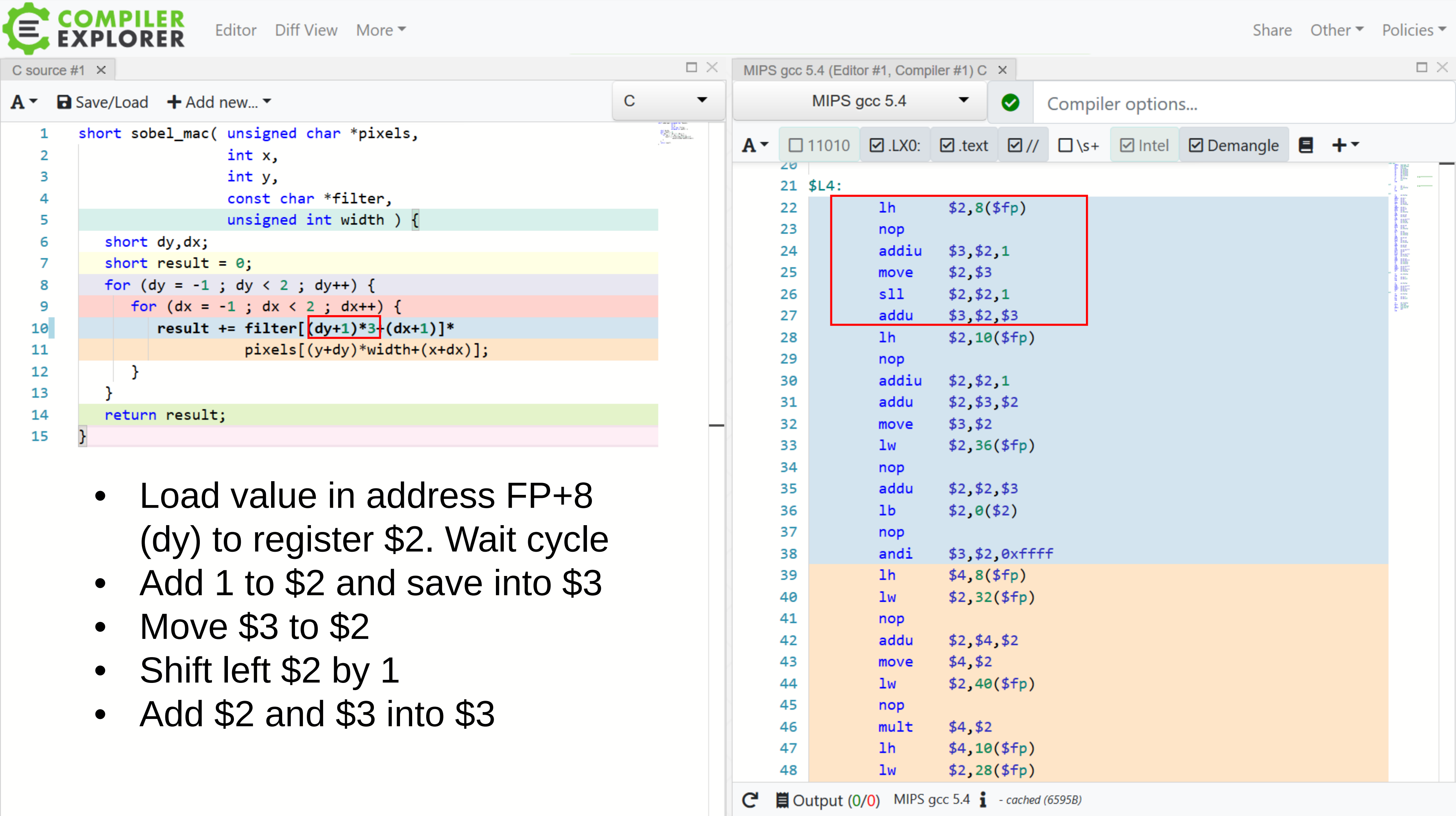
☐ 11010 ☑ .LX0: ☑ .text ☑ // ☐ \s+ ☑ Intel ☑ Demangle

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,10($fp)
$L3:
        lh      $2,10($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L4
        nop

        lh      $2,8($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,8($fp)
$L2:
        lh      $2,8($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L5
        nop

        lh      $2,12($fp)
        move    $sp,$fp
        lw      $fp,20($sp)
        addiu   $sp,$sp,24
        j       $31
        nop
```

- Load value in address FP+10 (dx) to register $2. Wait cycle
- Set $2 to 1 if $2 less than 2
- Branch to $L4 if $2 not 0
- Wait one cycle

Compiler explorer

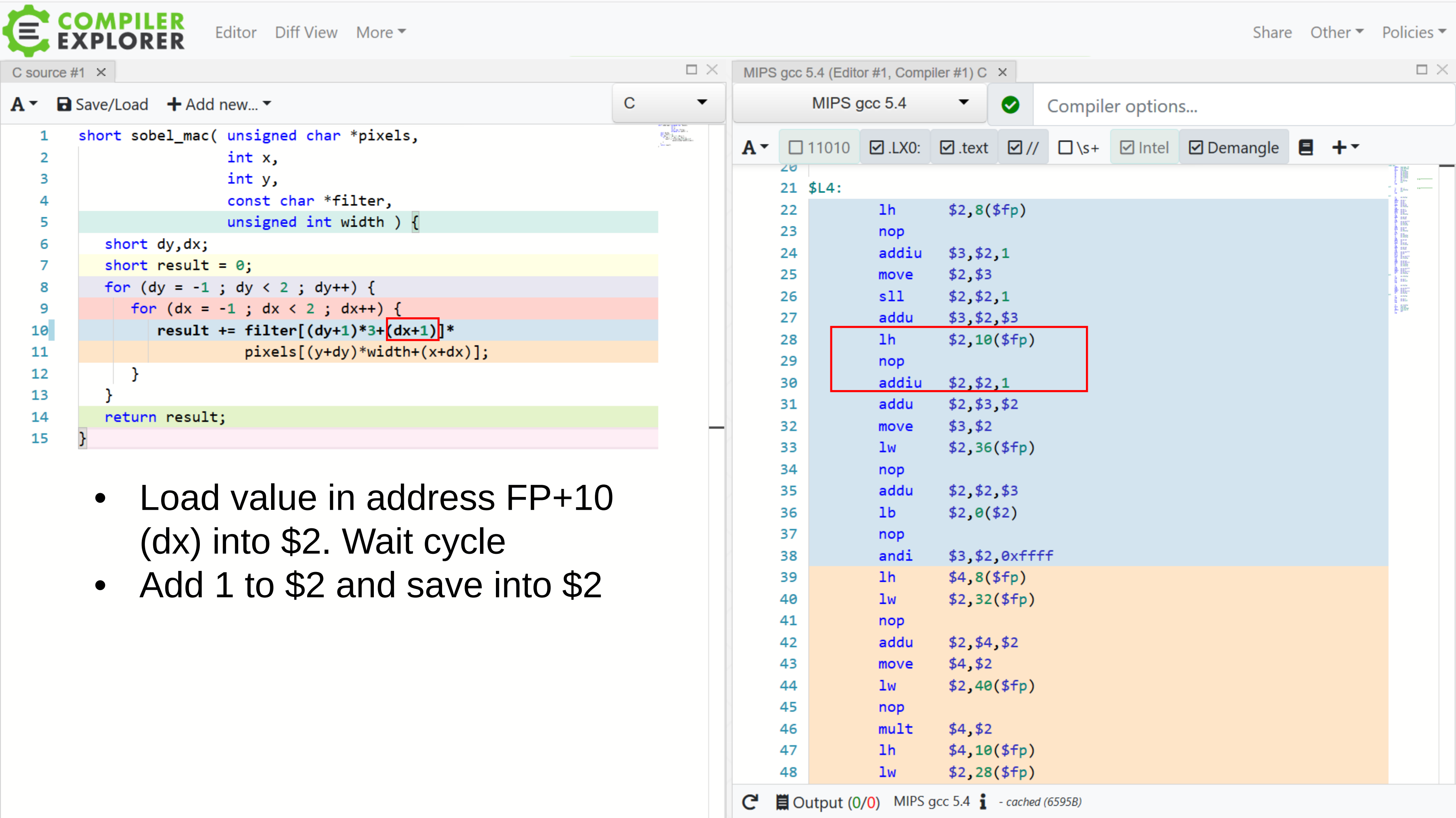


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```asm
$L4:
        lh      $2,8($fp)
        nop
        addiu   $3,$2,1
        move    $2,$3
        sll     $2,$2,1
        addu    $3,$2,$3
        lh      $2,10($fp)
        nop
        addiu   $2,$2,1
        addu    $2,$3,$2
        move    $3,$2
        lw      $2,36($fp)
        nop
        addu    $2,$2,$3
        lb      $2,0($2)
        nop
        andi    $3,$2,0xffff
        lh      $4,8($fp)
        lw      $2,32($fp)
        nop
        addu    $2,$4,$2
        move    $4,$2
        lw      $2,40($fp)
        nop
        mult    $4,$2
        lh      $4,10($fp)
        lw      $2,28($fp)
```

- Load value in address FP+8 (dy) to register $2. Wait cycle
- Add 1 to $2 and save into $3
- Move $3 to $2
- Shift left $2 by 1
- Add $2 and $3 into $3

Compiler explorer

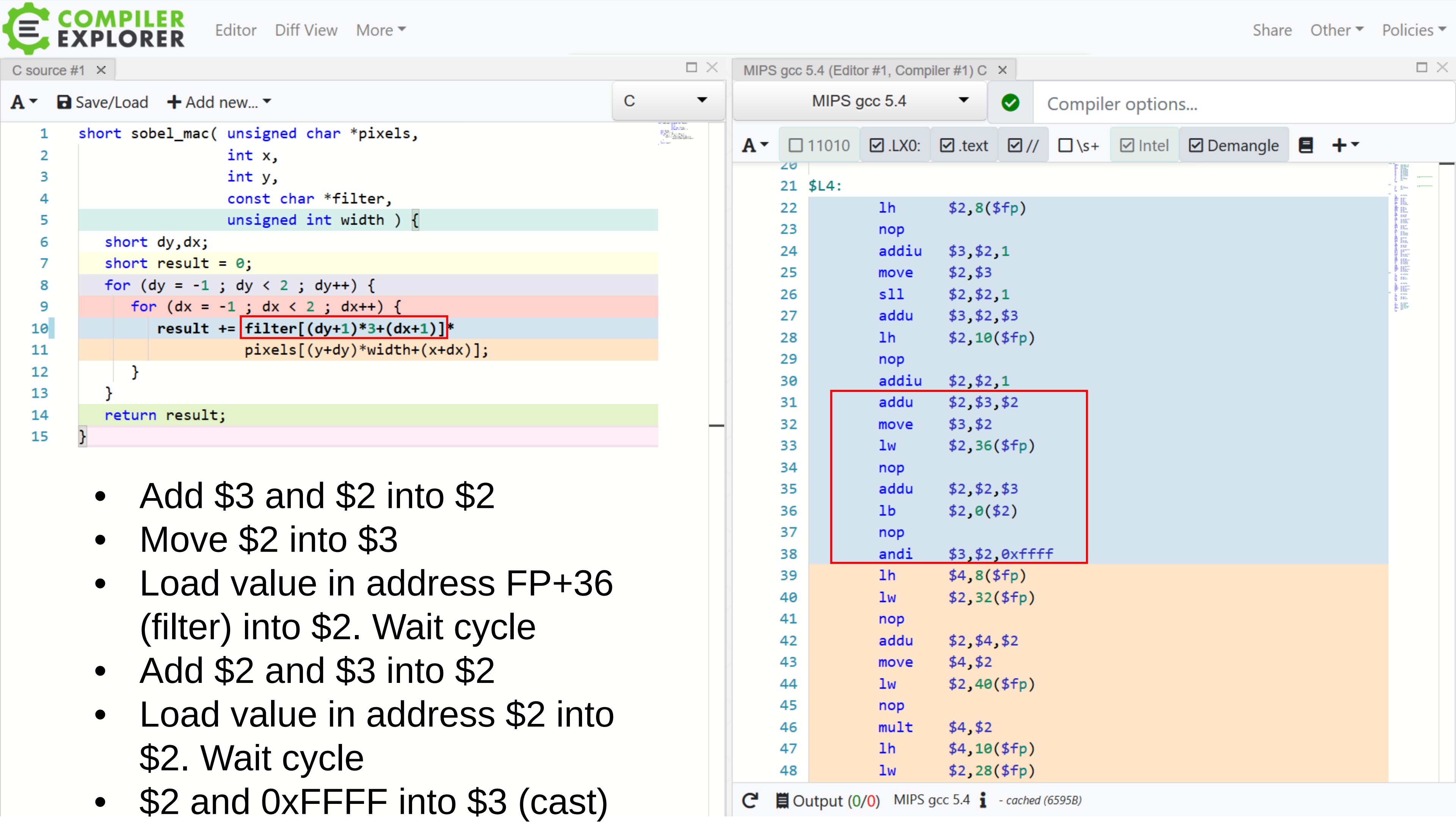


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
$L4:
        lh      $2,8($fp)
        nop
        addiu   $3,$2,1
        move    $2,$3
        sll     $2,$2,1
        addu    $3,$2,$3
        lh      $2,10($fp)
        nop
        addiu   $2,$2,1
        addu    $2,$3,$2
        move    $3,$2
        lw      $2,36($fp)
        nop
        addu    $2,$2,$3
        lb      $2,0($2)
        nop
        andi    $3,$2,0xffff
        lh      $4,8($fp)
        lw      $2,32($fp)
        nop
        addu    $2,$4,$2
        move    $4,$2
        lw      $2,40($fp)
        nop
        mult    $4,$2
        lh      $4,10($fp)
        lw      $2,28($fp)
```

- Load value in address FP+10 (dx) into $2. Wait cycle
- Add 1 to $2 and save into $2

Compiler explorer

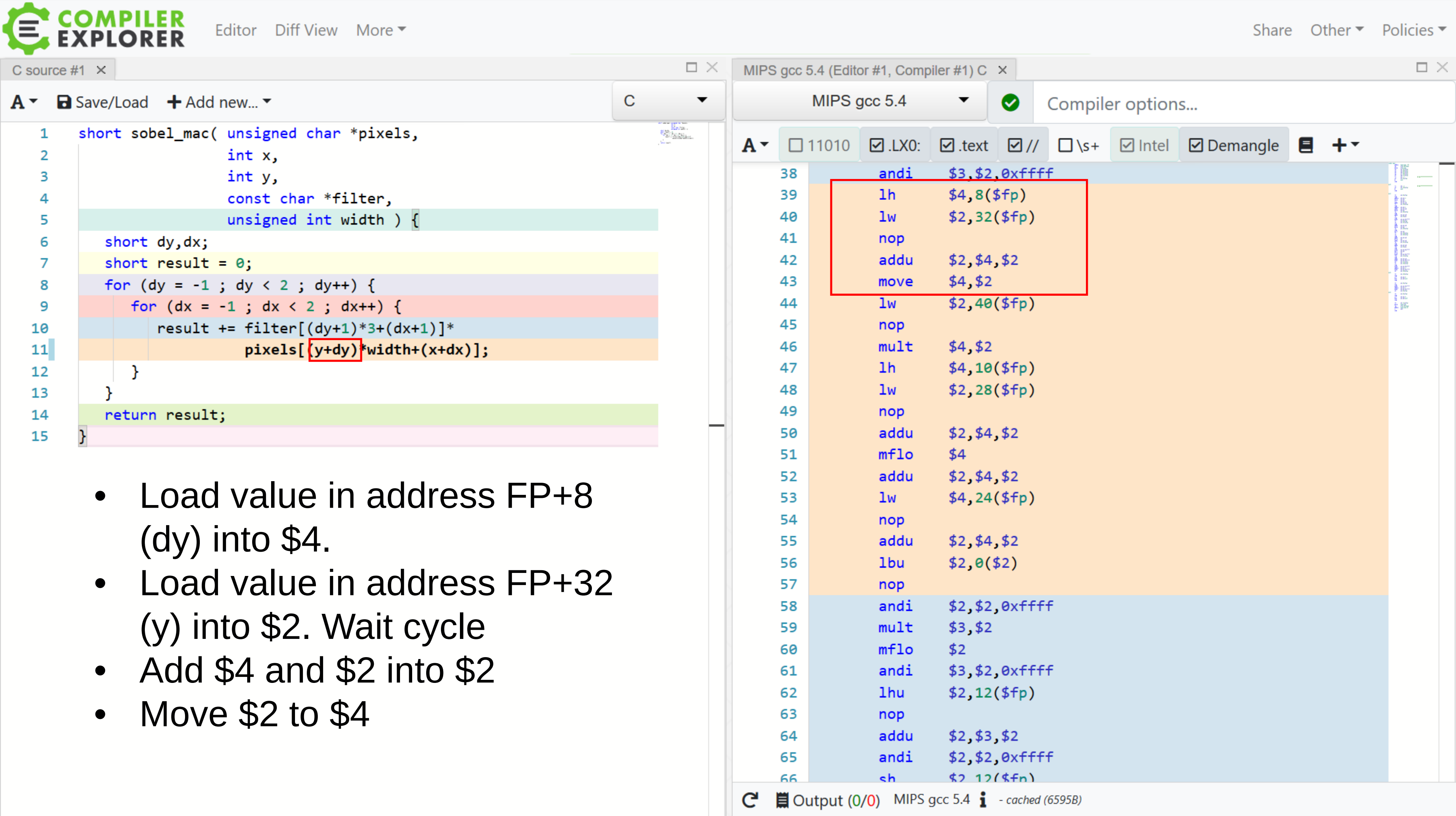


```
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
$L4:
        lh      $2,8($fp)
        nop
        addiu   $3,$2,1
        move    $2,$3
        sll     $2,$2,1
        addu    $3,$2,$3
        lh      $2,10($fp)
        nop
        addiu   $2,$2,1
        addu    $2,$3,$2
        move    $3,$2
        lw      $2,36($fp)
        nop
        addu    $2,$2,$3
        lb      $2,0($2)
        nop
        andi    $3,$2,0xffff
        lh      $4,8($fp)
        lw      $2,32($fp)
        nop
        addu    $2,$4,$2
        move    $4,$2
        lw      $2,40($fp)
        nop
        mult    $4,$2
        lh      $4,10($fp)
        lw      $2,28($fp)
```

- Add $3 and $2 into $2
- Move $2 into $3
- Load value in address FP+36 (filter) into $2. Wait cycle
- Add $2 and $3 into $2
- Load value in address $2 into $2. Wait cycle
- $2 and 0xFFFF into $3 (cast)

Compiler explorer

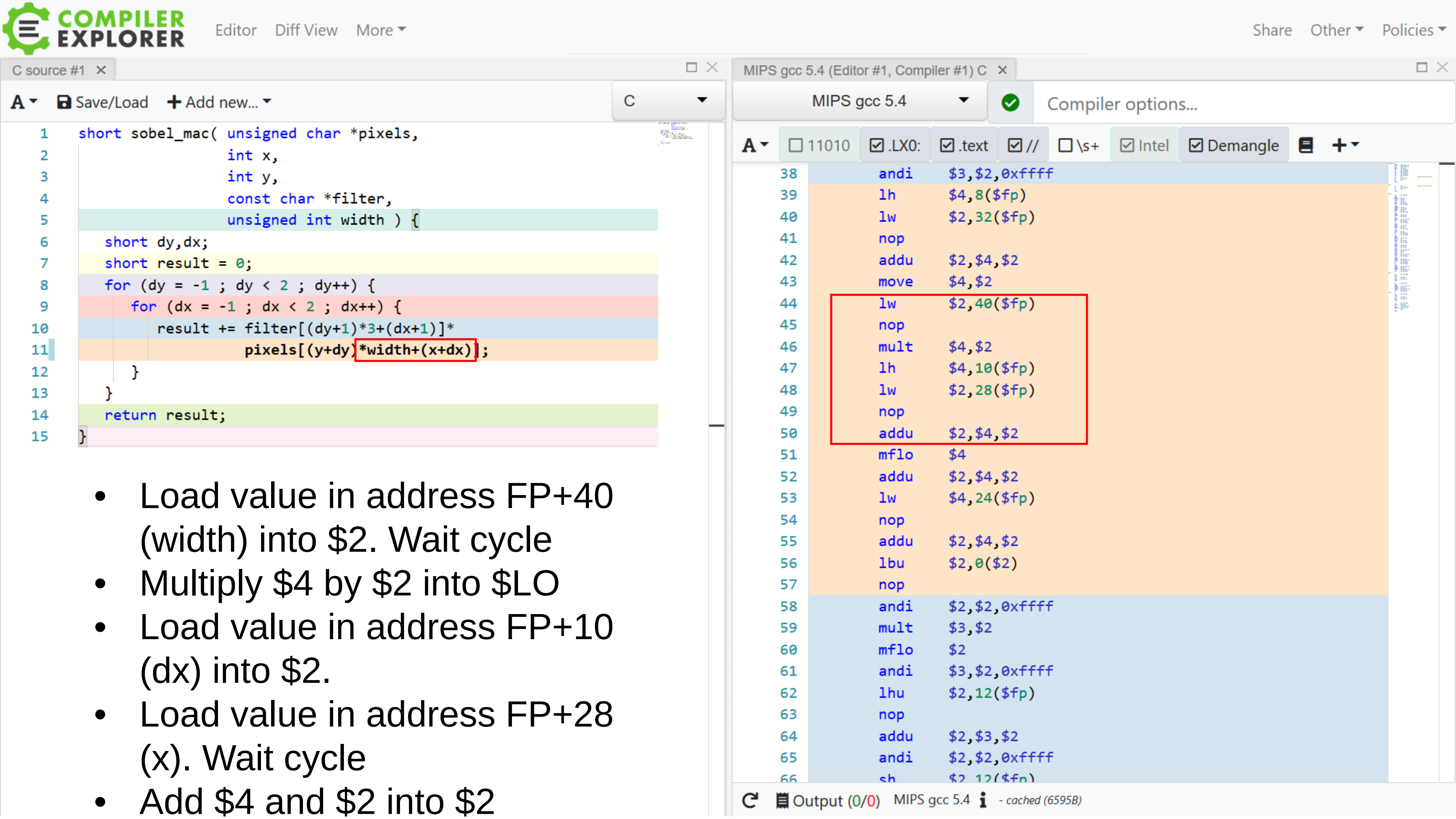


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
        andi    $3,$2,0xffff
        lh      $4,8($fp)
        lw      $2,32($fp)
        nop
        addu    $2,$4,$2
        move    $4,$2
        lw      $2,40($fp)
        nop
        mult    $4,$2
        lh      $4,10($fp)
        lw      $2,28($fp)
        nop
        addu    $2,$4,$2
        mflo    $4
        addu    $2,$4,$2
        lw      $4,24($fp)
        nop
        addu    $2,$4,$2
        lbu     $2,0($2)
        nop
        andi    $2,$2,0xffff
        mult    $3,$2
        mflo    $2
        andi    $3,$2,0xffff
        lhu     $2,12($fp)
        nop
        addu    $2,$3,$2
        andi    $2,$2,0xffff
        sh      $2,12($fp)
```

- Load value in address FP+8 (dy) into $4.
- Load value in address FP+32 (y) into $2. Wait cycle
- Add $4 and $2 into $2
- Move $2 to $4

Compiler explorer

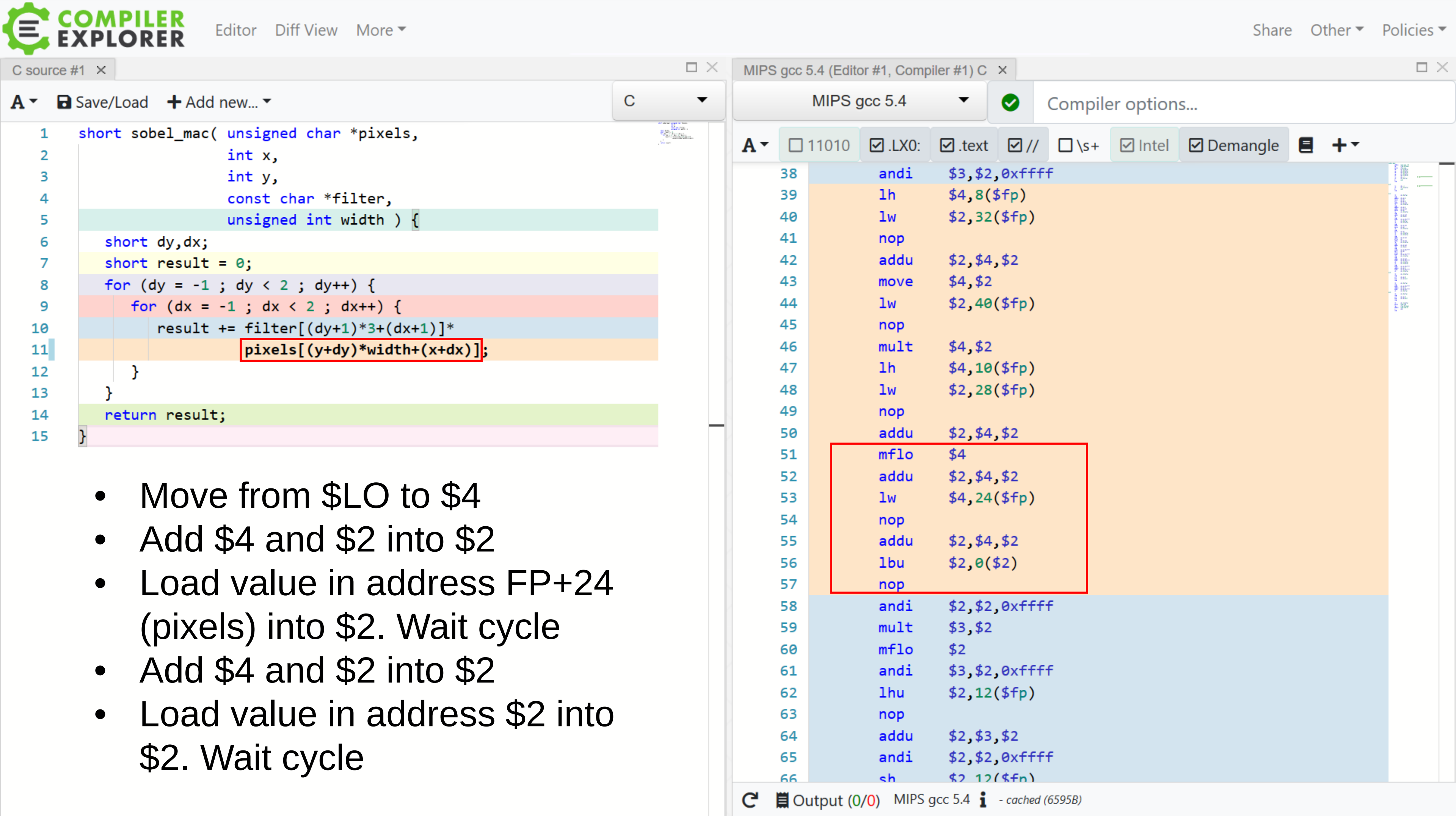


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
        andi    $3,$2,0xffff
        lh      $4,8($fp)
        lw      $2,32($fp)
        nop
        addu    $2,$4,$2
        move    $4,$2
        lw      $2,40($fp)
        nop
        mult    $4,$2
        lh      $4,10($fp)
        lw      $2,28($fp)
        nop
        addu    $2,$4,$2
        mflo    $4
        addu    $2,$4,$2
        lw      $4,24($fp)
        nop
        addu    $2,$4,$2
        lbu     $2,0($2)
        nop
        andi    $2,$2,0xffff
        mult    $3,$2
        mflo    $2
        andi    $3,$2,0xffff
        lhu     $2,12($fp)
        nop
        addu    $2,$3,$2
        andi    $2,$2,0xffff
        sh      $2,12($fp)
```

- Load value in address FP+40 (width) into $2. Wait cycle
- Multiply $4 by $2 into $LO
- Load value in address FP+10 (dx) into $2.
- Load value in address FP+28 (x). Wait cycle
- Add $4 and $2 into $2

Compiler explorer



C

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

MIPS gcc 5.4

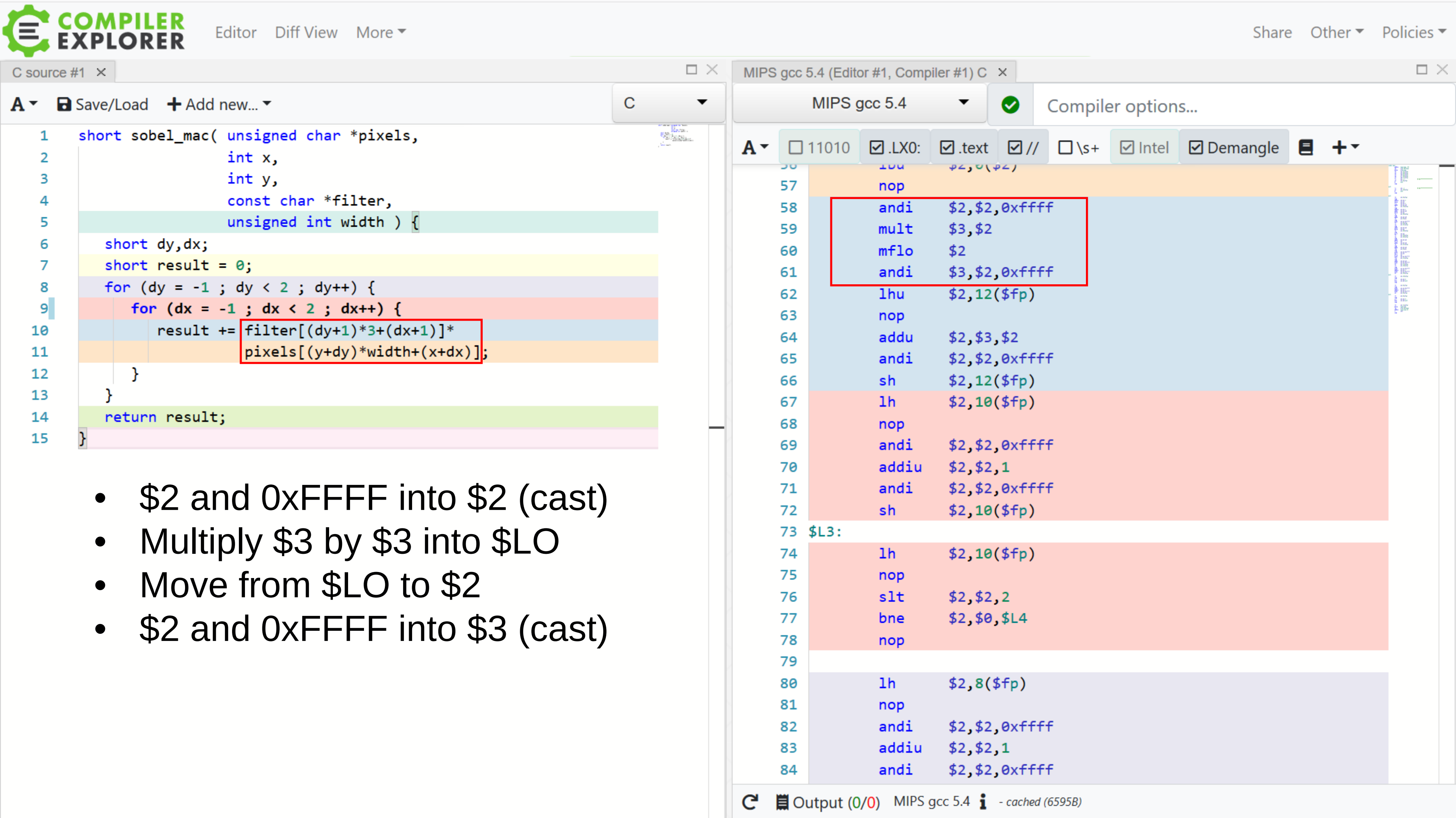
Compiler options...

□ 11010 ☑ .LX0: ☑ .text ☑ // □ \s+ ☑ Intel ☑ Demangle

```
        andi    $3,$2,0xffff
        lh      $4,8($fp)
        lw      $2,32($fp)
        nop
        addu    $2,$4,$2
        move    $4,$2
        lw      $2,40($fp)
        nop
        mult    $4,$2
        lh      $4,10($fp)
        lw      $2,28($fp)
        nop
        addu    $2,$4,$2
        mflo    $4
        addu    $2,$4,$2
        lw      $4,24($fp)
        nop
        addu    $2,$4,$2
        lbu     $2,0($2)
        nop
        andi    $2,$2,0xffff
        mult    $3,$2
        mflo    $2
        andi    $3,$2,0xffff
        lhu     $2,12($fp)
        nop
        addu    $2,$3,$2
        andi    $2,$2,0xffff
        sh      $2,12($fp)
```

- Move from $LO to $4
- Add $4 and $2 into $2
- Load value in address FP+24 (pixels) into $2. Wait cycle
- Add $4 and $2 into $2
- Load value in address $2 into $2. Wait cycle

Compiler explorer



Compiler: MIPS gcc 5.4 — Compiler options...

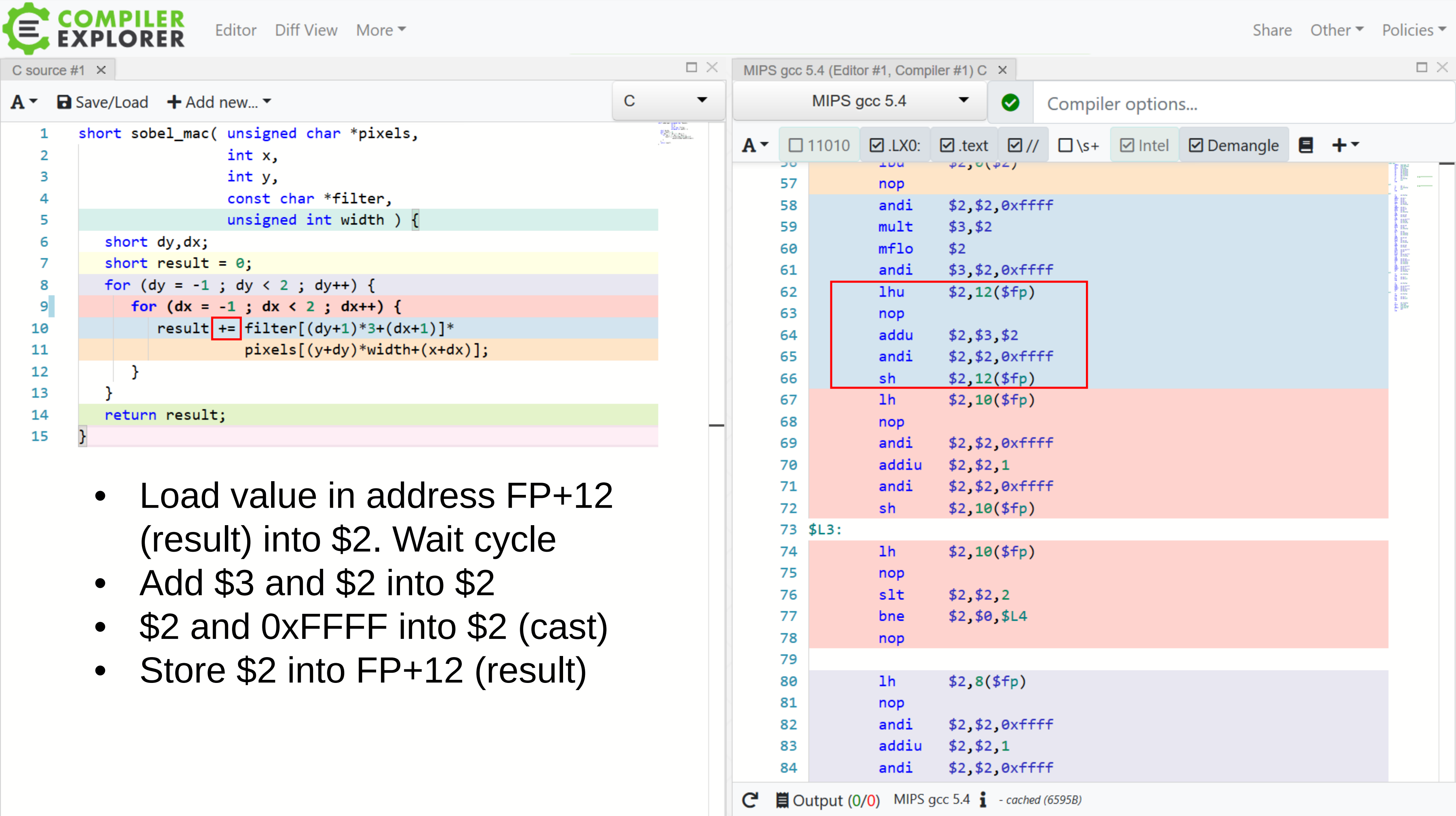
□ 11010 ☑ .LX0: ☑ .text ☑ // □ \s+ ☑ Intel ☑ Demangle

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
        nop
        andi    $2,$2,0xffff
        mult    $3,$2
        mflo    $2
        andi    $3,$2,0xffff
        lhu     $2,12($fp)
        nop
        addu    $2,$3,$2
        andi    $2,$2,0xffff
        sh      $2,12($fp)
        lh      $2,10($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,10($fp)
$L3:
        lh      $2,10($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L4
        nop

        lh      $2,8($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
```

- $2 and 0xFFFF into $2 (cast)
- Multiply $3 by $3 into $LO
- Move from $LO to $2
- $2 and 0xFFFF into $3 (cast)

Compiler explorer



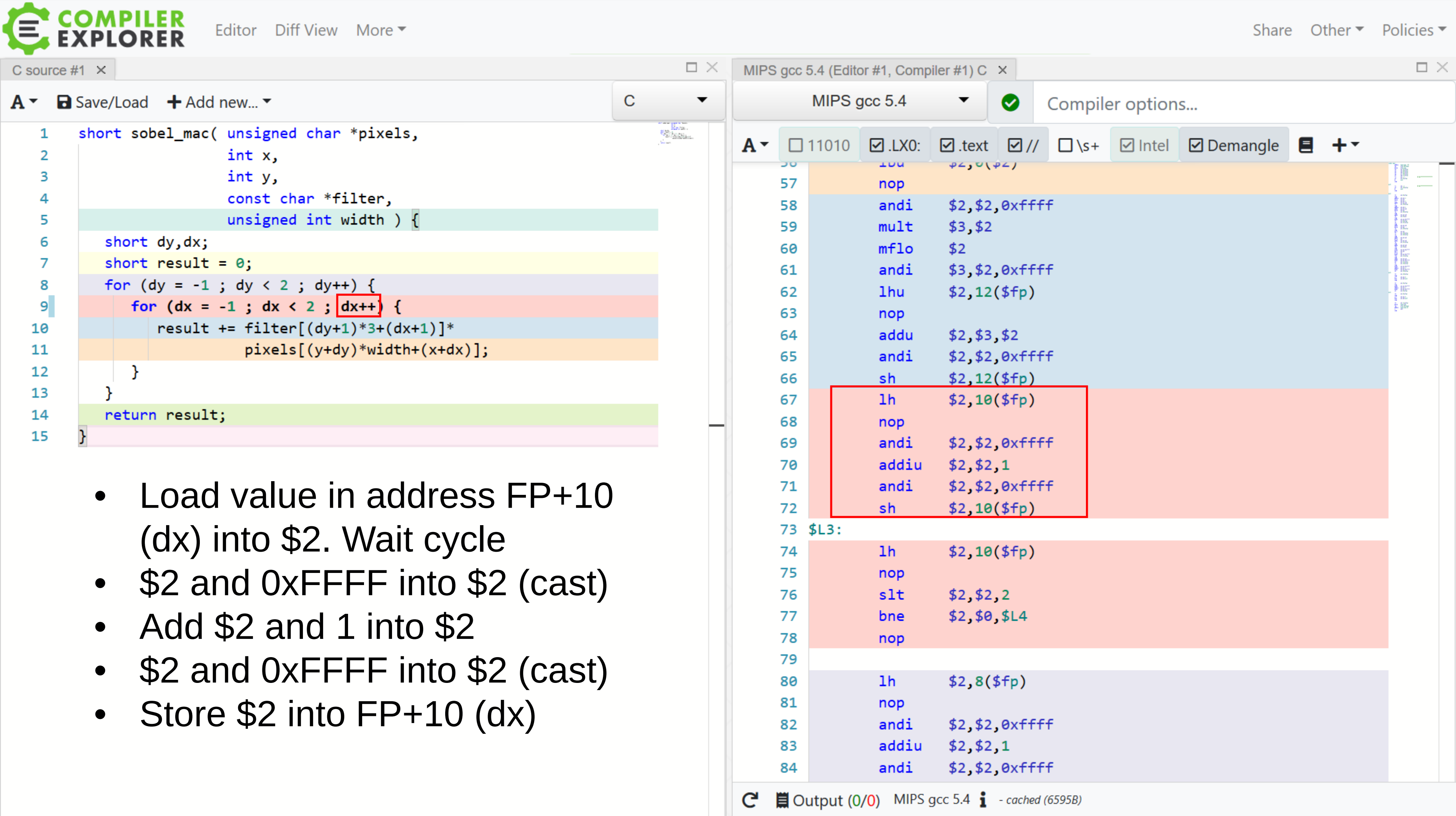
□ 11010 ☑ .LX0: ☑ .text ☑ // □ \s+ ☑ Intel ☑ Demangle

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
   short dy,dx;
   short result = 0;
   for (dy = -1 ; dy < 2 ; dy++) {
      for (dx = -1 ; dx < 2 ; dx++) {
         result += filter[(dy+1)*3+(dx+1)]*
                   pixels[(y+dy)*width+(x+dx)];
      }
   }
   return result;
}
```

```
        lbu     $2,0($2)
        nop
        andi    $2,$2,0xffff
        mult    $3,$2
        mflo    $2
        andi    $3,$2,0xffff
        lhu     $2,12($fp)
        nop
        addu    $2,$3,$2
        andi    $2,$2,0xffff
        sh      $2,12($fp)
        lh      $2,10($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,10($fp)
$L3:
        lh      $2,10($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L4
        nop

        lh      $2,8($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
```

- Load value in address FP+12 (result) into $2. Wait cycle
- Add $3 and $2 into $2
- $2 and 0xFFFF into $2 (cast)
- Store $2 into FP+12 (result)

Compiler explorer

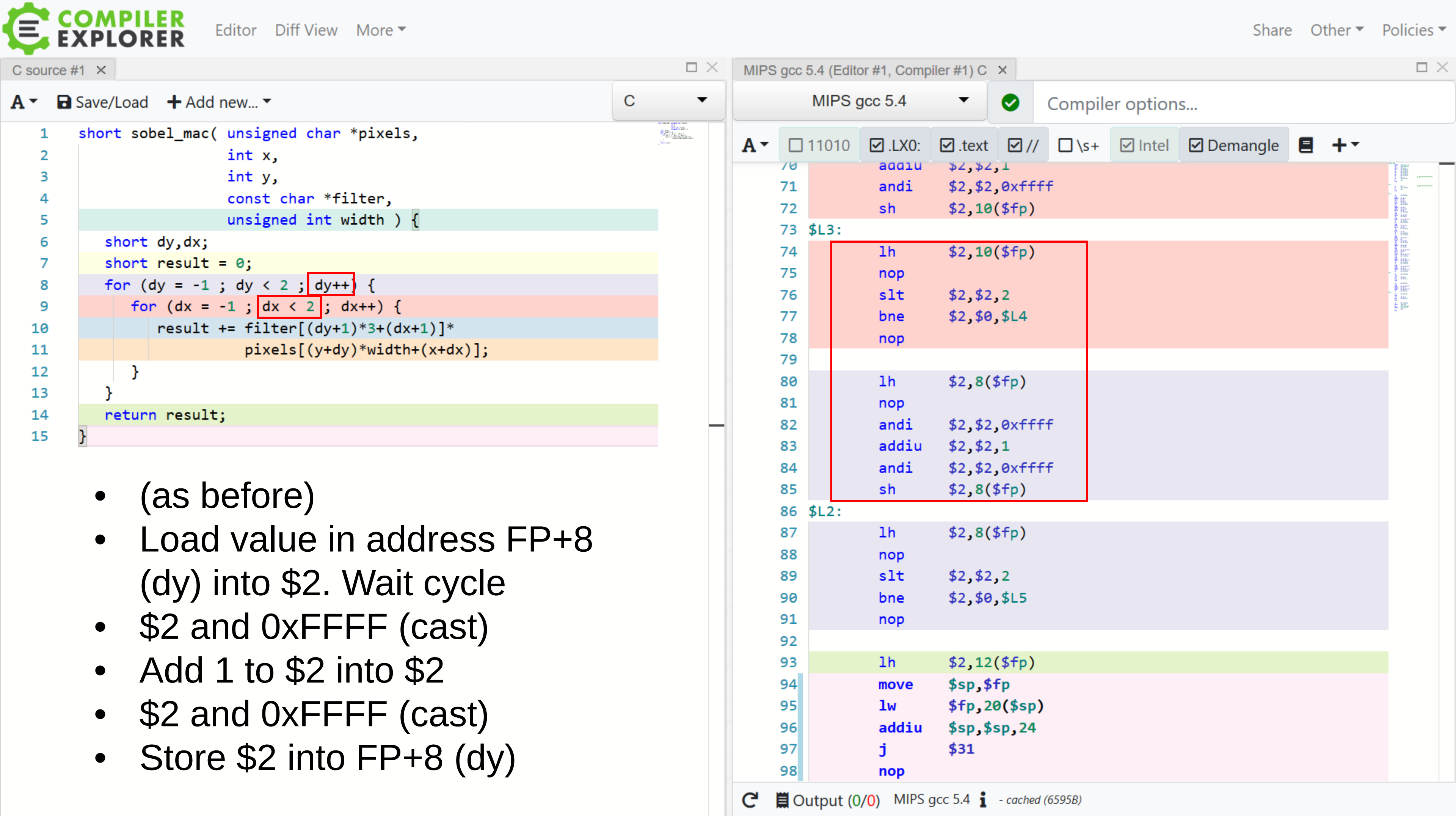


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
        lbu     $2,0($2)
        nop
        andi    $2,$2,0xffff
        mult    $3,$2
        mflo    $2
        andi    $3,$2,0xffff
        lhu     $2,12($fp)
        nop
        addu    $2,$3,$2
        andi    $2,$2,0xffff
        sh      $2,12($fp)
        lh      $2,10($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,10($fp)
$L3:
        lh      $2,10($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L4
        nop

        lh      $2,8($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
```

- Load value in address FP+10 (dx) into $2. Wait cycle
- $2 and 0xFFFF into $2 (cast)
- Add $2 and 1 into $2
- $2 and 0xFFFF into $2 (cast)
- Store $2 into FP+10 (dx)

Compiler explorer

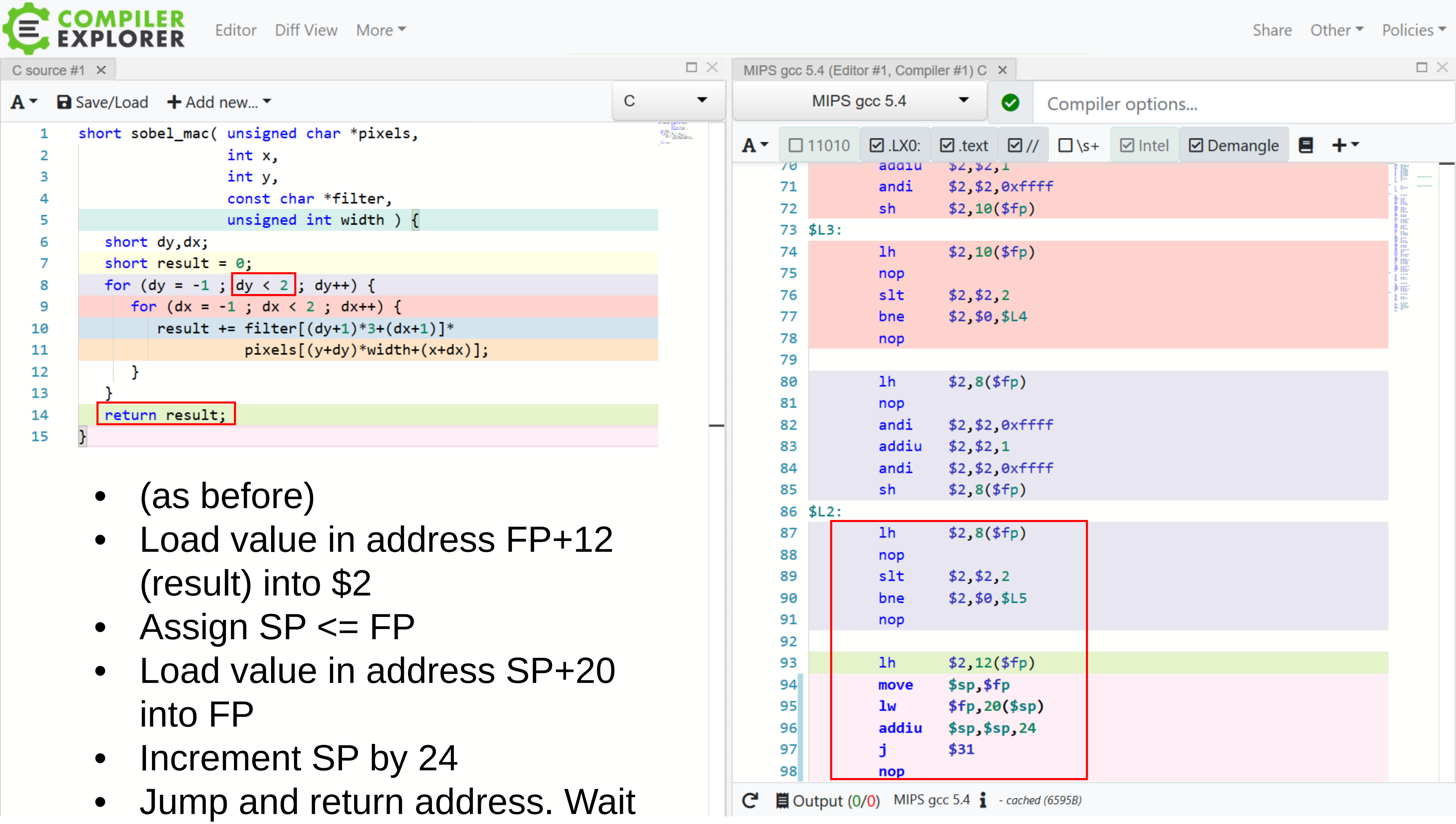


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

```
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,10($fp)
$L3:
        lh      $2,10($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L4
        nop

        lh      $2,8($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,8($fp)
$L2:
        lh      $2,8($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L5
        nop

        lh      $2,12($fp)
        move    $sp,$fp
        lw      $fp,20($sp)
        addiu   $sp,$sp,24
        j       $31
        nop
```

- (as before)
- Load value in address FP+8 (dy) into $2. Wait cycle
- $2 and 0xFFFF (cast)
- Add 1 to $2 into $2
- $2 and 0xFFFF (cast)
- Store $2 into FP+8 (dy)

Compiler explorer



```
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

MIPS gcc 5.4 — Compiler options...

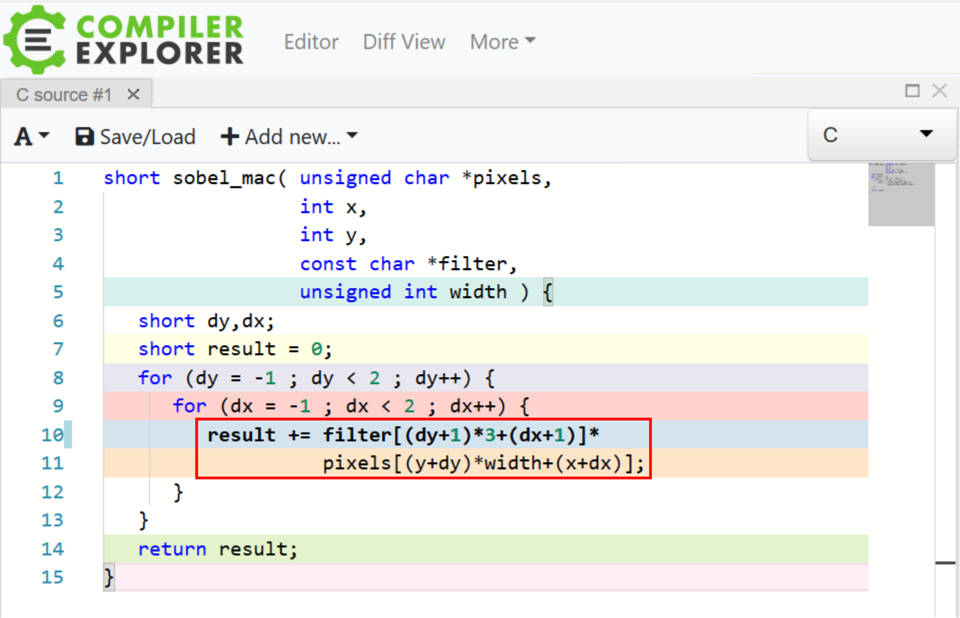
☐ 11010 ☑ .LX0: ☑ .text ☑ // ☐ \s+ ☑ Intel ☑ Demangle

```
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,10($fp)
$L3:
        lh      $2,10($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L4
        nop

        lh      $2,8($fp)
        nop
        andi    $2,$2,0xffff
        addiu   $2,$2,1
        andi    $2,$2,0xffff
        sh      $2,8($fp)
$L2:
        lh      $2,8($fp)
        nop
        slt     $2,$2,2
        bne     $2,$0,$L5
        nop

        lh      $2,12($fp)
        move    $sp,$fp
        lw      $fp,20($sp)
        addiu   $sp,$sp,24
        j       $31
        nop
```

Output (0/0) MIPS gcc 5.4 - cached (6595B)

- (as before)
- Load value in address FP+12 (result) into $2
- Assign SP <= FP
- Load value in address SP+20 into FP
- Increment SP by 24
- Jump and return address. Wait

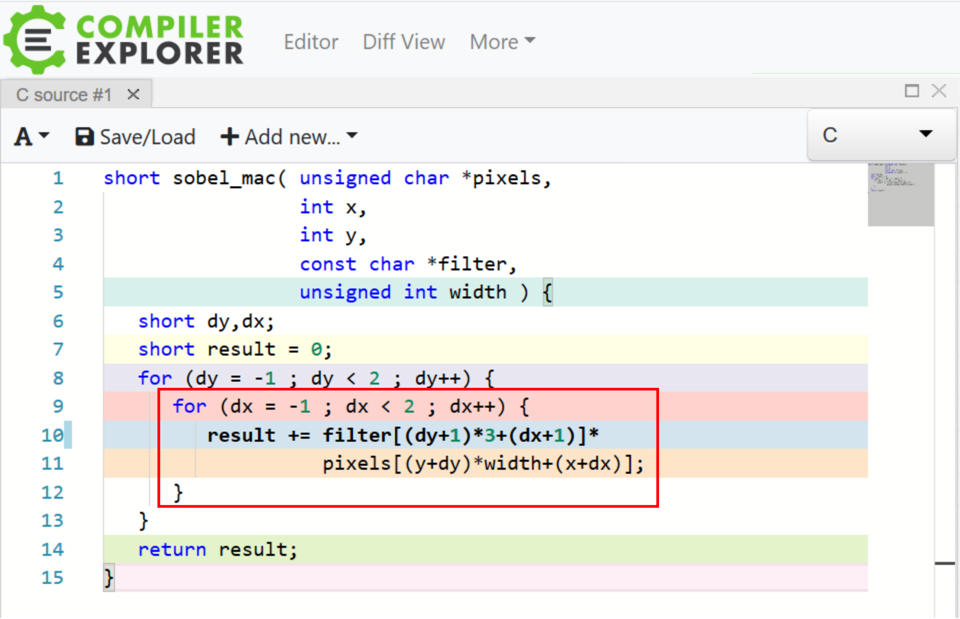
Compiler explorer



```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

Function	Cycles
Compute address filter	9
Filter value	8
Compute address pixels	14
Pixels value	5
Combine into result	9
TOTAL	**45**

Compiler explorer

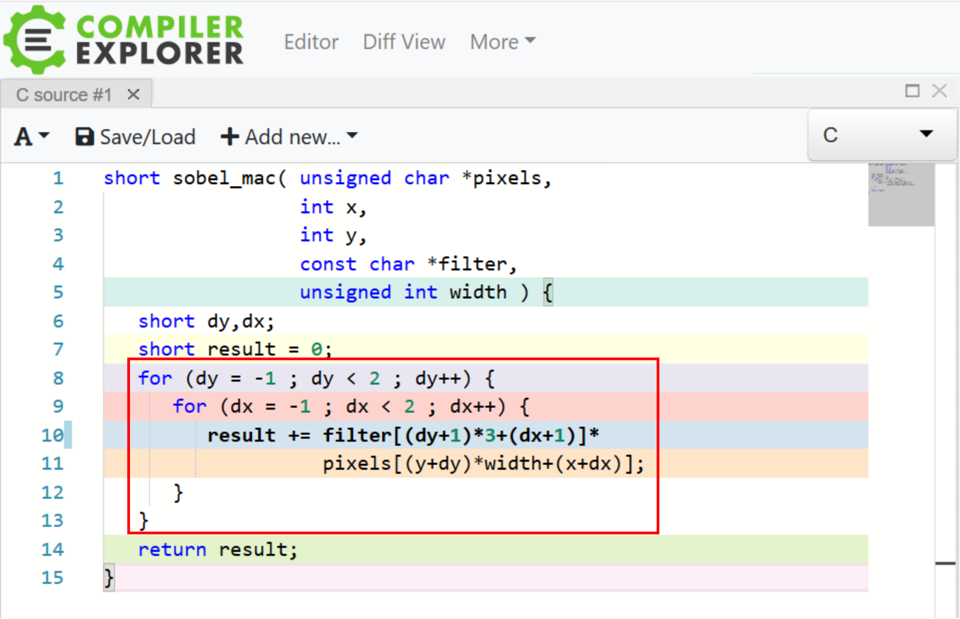


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
   short dy,dx;
   short result = 0;
   for (dy = -1 ; dy < 2 ; dy++) {
      for (dx = -1 ; dx < 2 ; dx++) {
         result += filter[(dy+1)*3+(dx+1)]*
                   pixels[(y+dy)*width+(x+dx)];
      }
   }
   return result;
}
```

Function	Cycles
Check loop dx	5*
Compute address filter	9
Filter value	8
Compute address pixels	14
Pixels value	5
Combine into result	9
Increment loop dx	6
TOTAL	**173***

(*) Branching

Compiler explorer

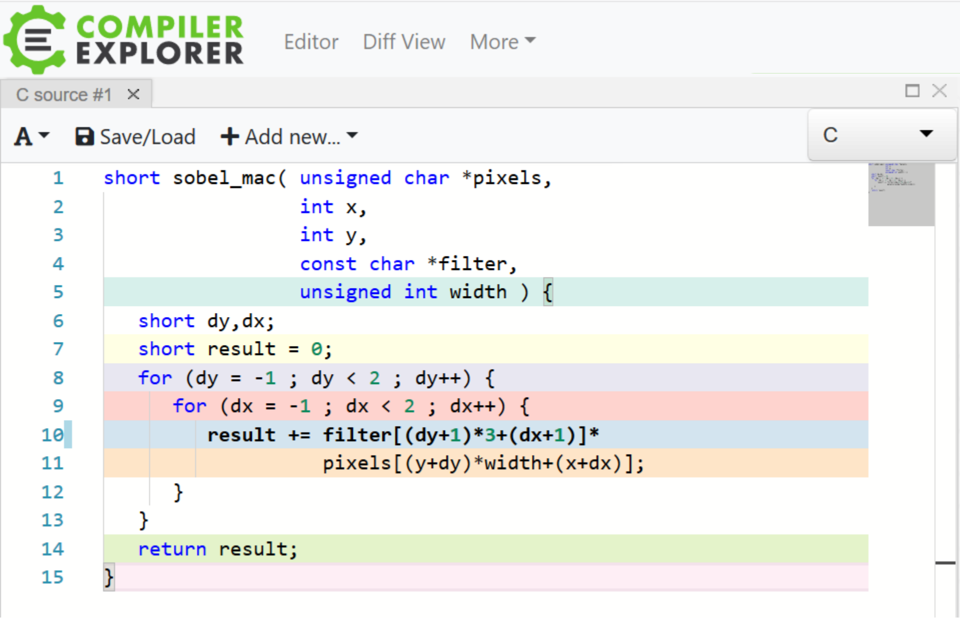


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

Function	Cycles
Check loop dy	5*
Check loop dx	5*
Compute address filter	9
Filter value	8
Compute address pixels	14
Pixels value	5
Combine into result	9
Increment loop dx	6
Increment loop dy	6
TOTAL	**557***

(*) Branching

Compiler explorer

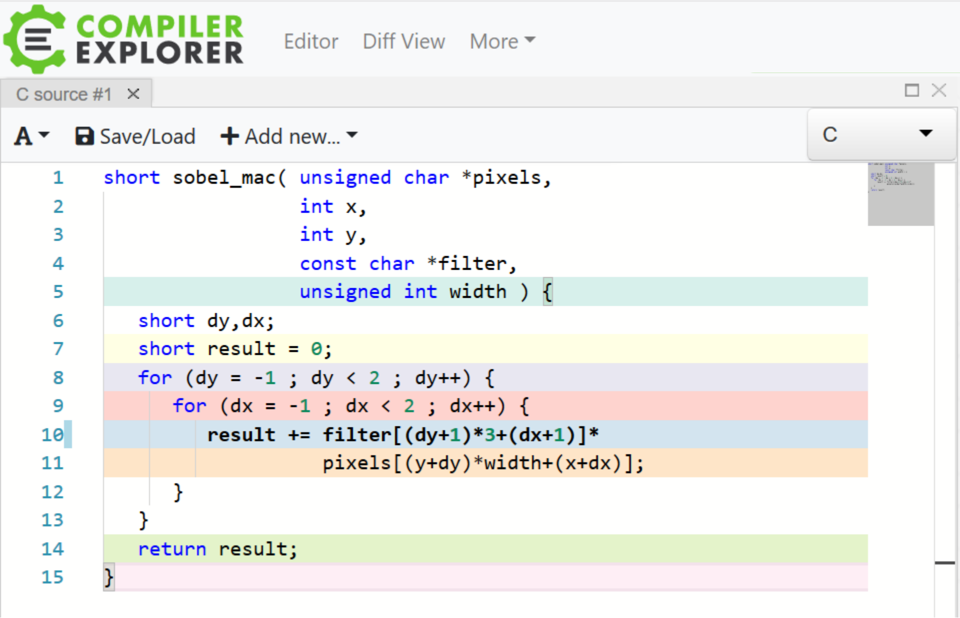


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
   short dy,dx;
   short result = 0;
   for (dy = -1 ; dy < 2 ; dy++) {
      for (dx = -1 ; dx < 2 ; dx++) {
         result += filter[(dy+1)*3+(dx+1)]*
                   pixels[(y+dy)*width+(x+dx)];
      }
   }
   return result;
}
```

Function	Cycles
Start function	7
Set initial result	1
Initialize loop dy	4*
Initialize loop dx	4*
Check loop dy	5*
Check loop dx	5*
Compute address filter	9
Filter value	8
Compute address pixels	14
Pixels value	5
Combine into result	9
Increment loop dx	6
Increment loop dy	6
Save result	1
End function	5
TOTAL	**579***

(*) Branching

Compiler explorer



```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

Function	Cycles
Start function	7
Set initial result	1
Initialize loop dy	4*
Initialize loop dx	4*
Check loop dy	5*
Check loop dx	5*
Compute address filter	9
Filter value	8
Compute address pixels	14
Pixels value	5
Combine into result	9
Increment loop dx	6
Increment loop dy	6
Save result	1
End function	5
194820x **TOTAL**	**579***

(*) Branching

Loop unrolling

- If we remove the inner loop?

Loop unrolling

- If we remove the inner loop?

Function	Cycles
Start function	7
Set initial result	1
Initialize loop dy	4*
Check loop dy	5*
Computing result	115
Increment loop dy	6
Save result	1
End function	5
TOTAL	**401***

Improvement: 579/401 = 1.44x

(*) Branching

Loop unrolling

- If we remove the inner loop?

Function	-O0	-O0 (dx removed)
conv_grayscale (Time)	127336549 (2.547 s)	127082760 (2.542 s)
sobel_x (Time)	729624798 (14.592 s)	551850939 (11.037 s)
sobel_y (Time)	729677739 (14.594 s)	551818999 (11.036 s)
sobel_threshold (Time)	102849150 (2.057 s)	105069463 (2.101 s)
TOTAL (Time)	**1689488236 (33.79 s)**	**1335822161 (26.72 s)**

1.32x

Function	Cycles
Start function	7
Set initial result	1
Initialize loop dy	4*
Check loop dy	5*
Computing result	115
Increment loop dy	6
Save result	1
End function	5
TOTAL	**401***

Improvement: 579/401 = 1.44x

(*) Branching

Loop unrolling

- If we remove the inner loop?

Function	-O0	-O0 (dx removed)
conv_grayscale (Time)	127336549 (2.547 s)	127082760 (2.542 s)
sobel_x (Time)	729624798 (14.592 s)	551850939 (11.037 s)
sobel_y (Time)	729677739 (14.594 s)	551818999 (11.036 s)
sobel_threshold (Time)	102849150 (2.057 s)	105069463 (2.101 s)
TOTAL (Time)	**1689488236 (33.79 s)**	**1335822161 (26.72 s)**

1.32x

1.26x

Function	Cycles
Start function	7
Set initial result	1
Initialize loop dy	4*
Check loop dy	5*
Computing result	115
Increment loop dy	6
Save result	1
End function	5
TOTAL	**401***

Improvement: 579/401 = 1.44x

(*) Branching

Loop unrolling

- If we remove the inner loop?
- If we remove the outer loop?
  - Improvement (MIPS): 579/275 = 2.11x

Function	-O0	-O0 (dx removed)	-O0 (no loops)
conv_grayscale (Time)	127336549 (2.547 s)	127082760 (2.542 s)	127071260 (2.541 s)
sobel_x (Time)	729624798 (14.592 s)	551850939 (11.037 s)	425314676 (8.506 s)
sobel_y (Time)	729677739 (14.594 s)	551818999 (11.036 s)	425378323 (8.508 s)
sobel_threshold (Time)	102849150 (2.057 s)	105069463 (2.101 s)	105635977 (2.113 s)
TOTAL (Time)	**1689488236 (33.79 s)**	**1335822161 (26.72 s)**	**1083400236 (21.67 s)**

1.72x

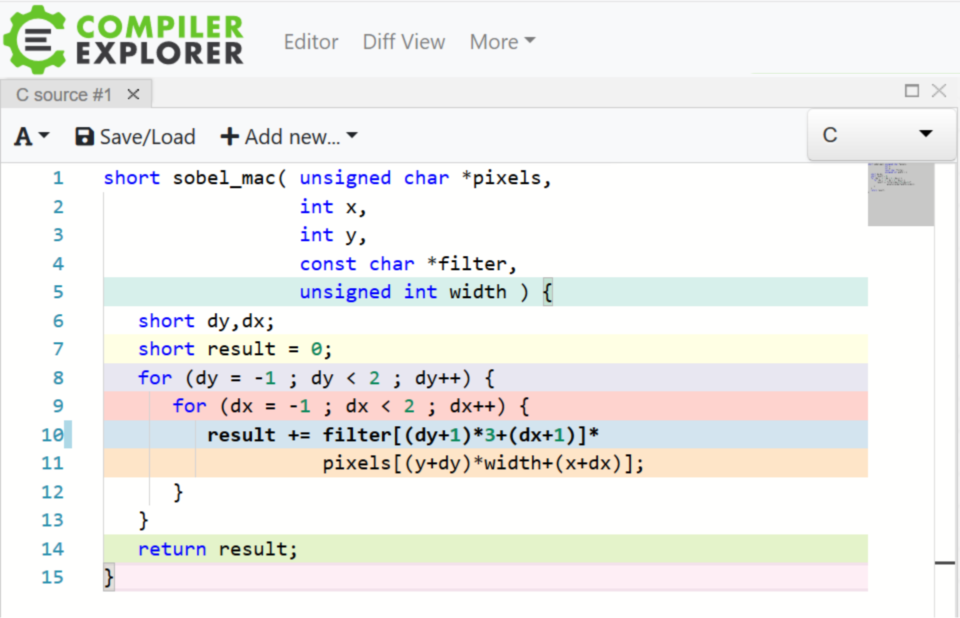
1.26x 1.56x

Loop unrolling

- Pros:
  - No need much knowledge of the algorithm
  - Save on overhead operations
  - Branch penalty

- Cons:
  - Improvement depends on the number of operations spent inside the loop
  - Only works when having a “perfect loop”: Number of iterations is predefined, does not depend on data
  - Worsen code readability

Can we do better?

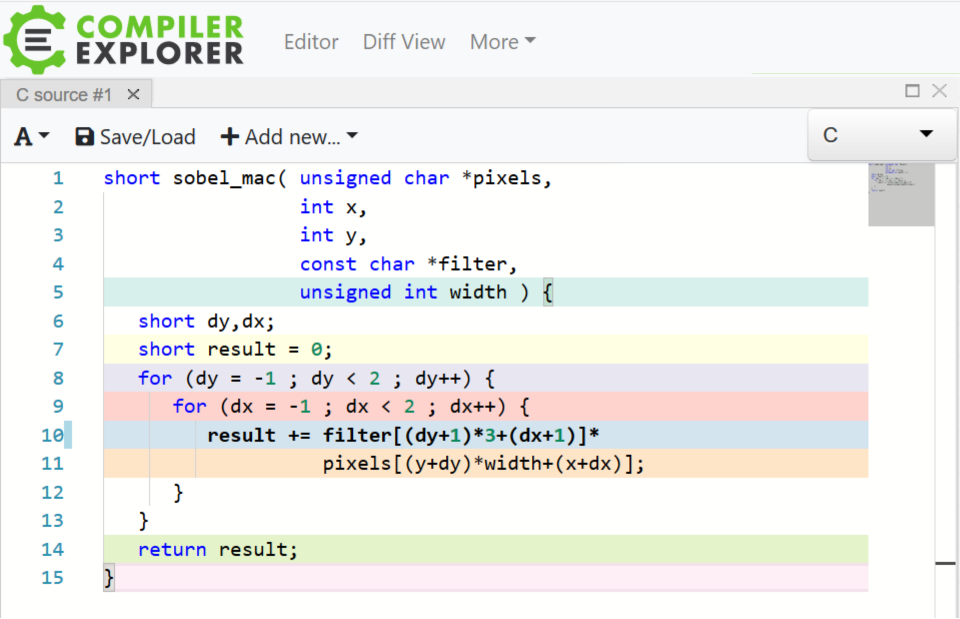


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
  short dy,dx;
  short result = 0;
  for (dy = -1 ; dy < 2 ; dy++) {
    for (dx = -1 ; dx < 2 ; dx++) {
      result += filter[(dy+1)*3+(dx+1)]*
                pixels[(y+dy)*width+(x+dx)];
    }
  }
  return result;
}
```

Function	Cycles
Start function	7
Set initial result	1
~~Initialize loop dy~~	~~4*~~
~~Initialize loop dx~~	~~4*~~
~~Check loop dy~~	~~5*~~
~~Check loop dx~~	~~5*~~
~~Compute address filter~~	~~9~~
~~Filter value~~	~~8~~
~~Compute address pixels~~	~~14~~
~~Pixels value~~	~~5~~
~~Combine into result~~	~~9~~
~~Increment loop dx~~	~~6~~
~~Increment loop dy~~	~~6~~
Save result	1
End function	5
TOTAL	**~~579*~~ 275**

(*) Branching

Can we do better?

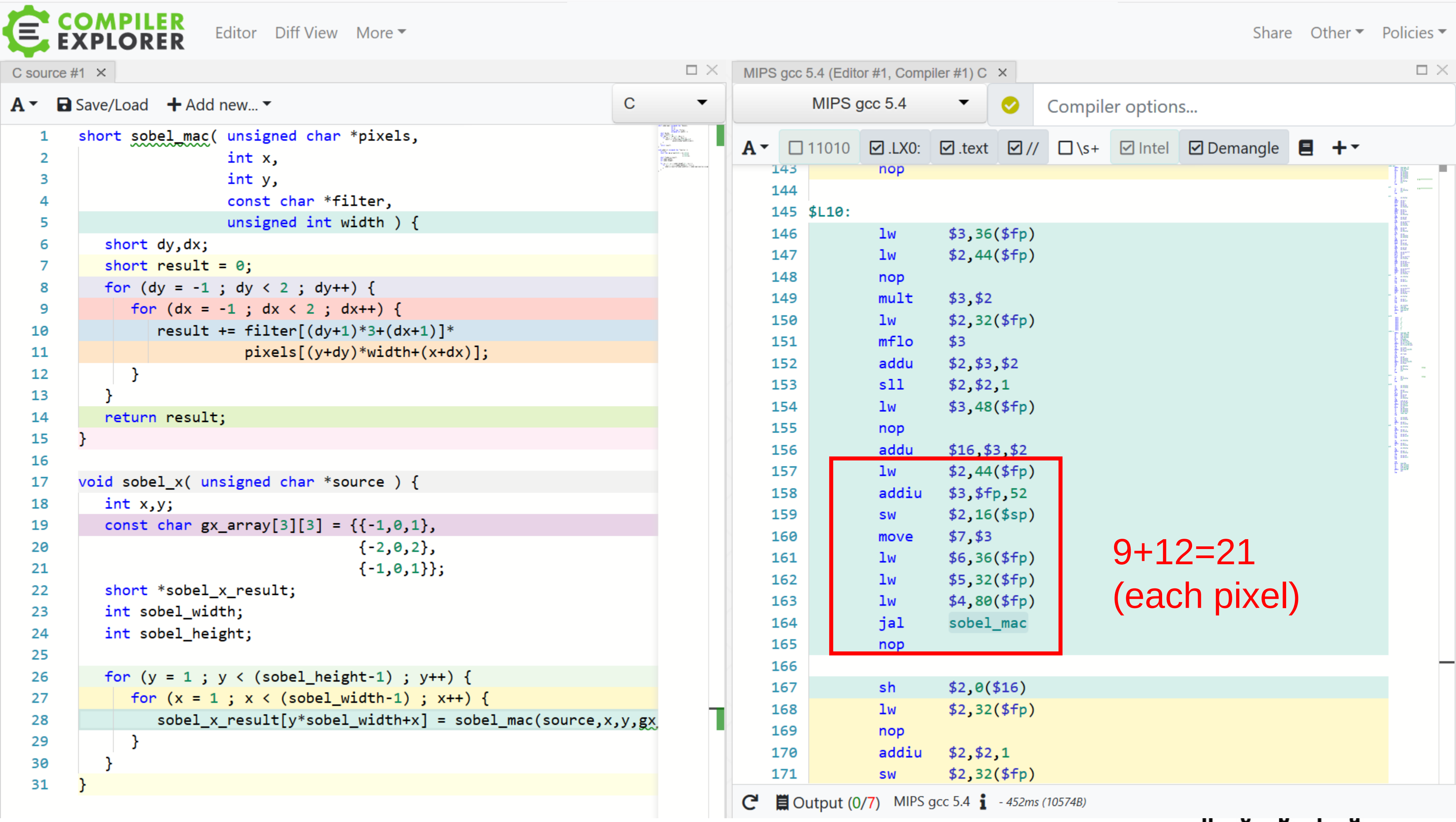


```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
   short dy,dx;
   short result = 0;
   for (dy = -1 ; dy < 2 ; dy++) {
      for (dx = -1 ; dx < 2 ; dx++) {
         result += filter[(dy+1)*3+(dx+1)]*
                   pixels[(y+dy)*width+(x+dx)];
      }
   }
   return result;
}
```

Function	Cycles
Start function	7
Set initial result	1
~~Initialize loop dy~~	~~4*~~
~~Initialize loop dx~~	~~4*~~
~~Check loop dy~~	~~5*~~
~~Check loop dx~~	~~5*~~
~~Compute address filter~~	~~9~~
~~Filter value~~	~~8~~
~~Compute address pixels~~	~~14~~
~~Pixels value~~	~~5~~
~~Combine into result~~	~~9~~
~~Increment loop dx~~	~~6~~
~~Increment loop dy~~	~~6~~
Save result	1
End function	5
TOTAL	**~~579*~~ 275**

(*) Branching

Can we do better?



MIPS gcc 5.4

☐ 11010 ☑ .LX0: ☑ .text ☑ // ☐ \s+ ☑ Intel ☑ Demangle

```c
short sobel_mac( unsigned char *pixels,
                 int x,
                 int y,
                 const char *filter,
                 unsigned int width ) {
   short dy,dx;
   short result = 0;
   for (dy = -1 ; dy < 2 ; dy++) {
      for (dx = -1 ; dx < 2 ; dx++) {
         result += filter[(dy+1)*3+(dx+1)]*
                   pixels[(y+dy)*width+(x+dx)];
      }
   }
   return result;
}

void sobel_x( unsigned char *source ) {
   int x,y;
   const char gx_array[3][3] = {{-1,0,1},
                                {-2,0,2},
                                {-1,0,1}};

   short *sobel_x_result;
   int sobel_width;
   int sobel_height;

   for (y = 1 ; y < (sobel_height-1) ; y++) {
      for (x = 1 ; x < (sobel_width-1) ; x++) {
         sobel_x_result[y*sobel_width+x] = sobel_mac(source,x,y,gx
      }
   }
}
```

```asm
        nop

$L10:
        lw      $3,36($fp)
        lw      $2,44($fp)
        nop
        mult    $3,$2
        lw      $2,32($fp)
        mflo    $3
        addu    $2,$3,$2
        sll     $2,$2,1
        lw      $3,48($fp)
        nop
        addu    $16,$3,$2
        lw      $2,44($fp)
        addiu   $3,$fp,52
        sw      $2,16($sp)
        move    $7,$3
        lw      $6,36($fp)
        lw      $5,32($fp)
        lw      $4,80($fp)
        jal     sobel_mac
        nop

        sh      $2,0($16)
        lw      $2,32($fp)
        nop
        addiu   $2,$2,1
        sw      $2,32($fp)
```

9+12=21
(each pixel)

Output (0/7) MIPS gcc 5.4 - 452ms (10574B)

In-lining

- Removing the function

Function	-O0	-O0 (no loops)	-O0 (no loops, inline)
conv_grayscale (Time)	127336549 (2.547 s)	127071260 (2.541 s)	125935754 (2.519 s)
sobel_x (Time)	729624798 (14.592 s)	425314676 (8.506 s)	325845850 (6.517 s)
sobel_y (Time)	729677739 (14.594 s)	425378323 (8.508 s)	320545780 (6.411 s)
sobel_threshold (Time)	102849150 (2.057 s)	105635977 (2.113 s)	104956605 (2.099 s)
TOTAL (Time)	**1689488236 (33.79 s)**	**1083400236 (21.67 s)**	**877283989 (17.55 s)**

1.31x
(2.24x)

1.56x

1.93x