Workshop on Essential Abstractions in GCC

Incremental Machine Descriptions for Spim: Levels 2, 3, and 4

GCC Resource Center

(www.cse.iitb.ac.in/grc)

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

July 2009

Outline

- Constructs supported in level 2
- Constructs supported in level 3
- Constructs supported in level 4 (left as exercises in the lab)

Part 1

Constructs Supported in Level 2

Arithmetic Operations Required in Level 2

Operation	Primitive Variants	Implementation	Remark
$Dest \leftarrow Src_1 - Src_2$	$R_i \leftarrow R_j - R_k$	`sub ri, rj, rk`	level 2
$Dest \leftarrow -Src$	$R_i \leftarrow -R_j$	`neg ri, rj`	
$Dest \leftarrow Src_1/Src_2$	$R_i \leftarrow R_j/R_k$	`div rj, rk` `mflo ri`	
$Dest \leftarrow Src_1\%Src_2$	$R_i \leftarrow R_j\%R_k$	`rem ri, rj, rk`	
$Dest \leftarrow Src_1 * Src_2$	$R_i \leftarrow R_j * R_k$	`mul ri, rj, rk`	

Notes

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 2/25
Arithmetic Operations Required in Level 2
Notes
Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Bitwise Operations Required in Level 2

Operation	Primitive Variants	Implementation	Remark
$Dest \leftarrow Src_1 \ll Src_2$	$R_i \leftarrow R_j \ll R_k$	`sllv ri, rj, rk`	level 2
	$R_i \leftarrow R_j \ll C_5$	`sll ri, rj, c`	
$Dest \leftarrow Src_1 \gg Src_2$	$R_i \leftarrow R_j \gg R_k$	`srav ri, rj, rk`	
	$R_i \leftarrow R_j \gg C_5$	`sra ri, rj, c`	
$Dest \leftarrow Src_1 \& Src_2$	$R_i \leftarrow R_j \& R_k$	`and ri, rj, rk`	
	$R_i \leftarrow R_j \& C$	`andi ri, rj, c`	
$Dest \leftarrow Src_1 \mid Src_2$	$R_i \leftarrow R_j \mid R_k$	`or ri, rj, rk`	
	$R_i \leftarrow R_j \mid C$	`ori ri, rj, c`	
$Dest \leftarrow Src_1 \hat{} Src_2$	$R_i \leftarrow R_j \hat{} R_k$	`xor ri, rj, rk`	
	$R_i \leftarrow R_j \hat{} C$	`xori ri, rj, c`	
$Dest \leftarrow \sim Src$	$R_i \leftarrow \sim R_j$	`not ri, rj`	

Notes

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 3/25
Bitwise Operations Required in Level 2
Notes
Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Divide Operation in `spim2.md` using define_insn

- For division, the spim architecture imposes use of multiple asm instructions for single operation.
- Two ASM instructions are emitted using single RTL pattern

```
(define_insn "divsi3"
 [(set (match_operand:SI 0 "register_operand" "=r")
       (div:SI (match_operand:SI 1 "register_operand" "r")
               (match_operand:SI 2 "register_operand" "r"))
 )]
 ""
 "div\\t%1, %2\\n\\tmflo\\t%0"
)
```


July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 4/25

Divide Operation in `spim2.md` using define_insn

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Advantages/Disadvantages of using define_insn

- Very simple to add the pattern
- Primitive target feature represented as single insn pattern in `.md`
- Unnecessary atomic grouping of instructions
- May hamper optimizations in general, and instruction scheduling, in particluar

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 5/25

Advantages/Disadvantages of using define_insn

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Divide Operation in spim2.md using define_expand

- The RTL pattern can be expanded into two different RTLs.

```
(define_expand "divsi3"
 [(parallel[(set (match_operand:SI 0 "register_operand" "")
       (div:SI (match_operand:SI 1 "register_operand" "")
               (match_operand:SI 2 "register_operand" "")))
  )
  (clobber (reg:SI 26))
  (clobber (reg:SI 27))])]
  ""

    emit_insn(gen_IITB_divide(gen_rtx_REG(SImode,26),
                              operands[1],
      operands[2]));
    emit_insn(gen_IITB_move_from_lo(operands[0],
                                    gen_rtx_REG(SImode,26)));
    DONE;

  )
```

Notes

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 6/25

Divide Operation in spim2.md using define_expand

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Divide Operation in spim2.md using define_expand

- Divide pattern equivalent to div instruction in architecture.

```
(define_insn "IITB_divide"
 [(parallel[(set (match_operand:SI 0 "LO_register_operand" "=q")
       (div:SI (match_operand:SI 1 "register_operand" "r")
               (match_operand:SI 2 "register_operand" "r"))
 )
 (clobber (reg:SI 27))])]
 ""
 "div t%1, %2"
)
```

Notes

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 7/25

Divide Operation in spim2.md using define_expand

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Divide Operation in spim2.md using define_expand

- Moving contents of special purpose register LO to/from general purpose register

```
(define_insn "IITB_move_from_lo"
 [(set (match_operand:SI 0 "register_operand" "=r")
       (match_operand:SI 1 "LO_register_operand" "q"))]
 ""
 "mflo \\t%0"
)

(define_insn "IITB_move_to_lo"
 [(set (match_operand:SI 0 "LO_register_operand" "=q")
       (match_operand:SI 1 "register_operand" "r"))]
 ""
 "mtlo \\t%1"
)
```

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 8/25

Divide Operation in spim2.md using define_expand

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 9/25

Divide Operation in spim2.md using define_expand

- Divide pattern equivalent to div instruction in architecture.

```
(define_insn "modsi3"
 [(parallel[(set (match_operand:SI 0 "register_operand" "=r")
       (mod:SI (match_operand:SI 1 "register_operand" "r")
               (match_operand:SI 2 "register_operand" "r"))
 )
 (clobber (reg:SI 26))
 (clobber (reg:SI 27))])]
 ""
 "rem \t%0, %1, %2"
)
```

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 9/25

Divide Operation in spim2.md using define_expand

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Advantages/Disadvantages of using define_expand

- Two instructions are seperated out at GIMPLE to RTL conversion phase
- Both instructions can undergo all RTL optimizations independently
- C interface is needed in md
- compilation becomes slower and requires more space

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 10/25

Advantages/Disadvantages of using define_expand

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Divide Operation in `spim2.md` using define_split

```
(define_split
 [(parallel [(set (match_operand:SI 0 "register_operand" "")
    (div:SI (match_operand:SI 1 "register_operand" "")
       (match_operand:SI 2 "register_operand" "")))
    )
    (clobber (reg:SI 26))
    (clobber (reg:SI 27))])]
    ""
 [(parallel [(set (match_dup 3)
    (div:SI (match_dup 1)
       (match_dup 2)))
    (clobber (reg:SI 27))])
    (set (match_dup 0)
      (match_dup 3))
 ]
        "operands[3]=gen_rtx_REG(SImode,26); "
)
```

```
[(parallel[
  (set (match_operand:SI 0 "LO_register_operand" "=q")
   (div:SI (match_operand:SI 1 "register_operand" "r")
    (match_operand:SI 2 "register_operand" "r")))
  (clobber (reg:SI 27))])]
```

```
[(set (match_operand:SI 0 "register_operand" "=r")
  (match_operand:SI 1 "LO_register_operand" "q"))]
```

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 2 11/25

Divide Operation in spim2.md using define_split

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Part 2

Constructs Supported in Level 3

Operations Required in Level 3

Operation	Primitive Variants	Implementation	Remark
$Dest \leftarrow fun(P_1, \ldots, P_n)$	`call` L_{fun}, n	`lw` r_i, [SP] `sw` r_i, [SP] : `lw` r_i, [SP-n*4] `sw` r_i, [SP-n*4]	Level 1
		jal L	New
		$Dest \leftarrow \$v0$	level 1
$fun(P_1, P_2, \ldots, P_n)$	`call` L_{fun}, n	`lw` r_i, [SP] `sw` r_i, [SP] : `lw` r_i, [SP-n*4] `sw` r_i, [SP-n*4]	Level 1
		jal L	New

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 3 12/25

Operations Required in Level 3

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Call Operation in spim3.md

```
(define_insn "call"
 [(call (match_operand:SI 0 "memory_operand" "m")
        (match_operand:SI 1 "immediate_operand" "i"))
  (clobber (reg:SI 31))
 ]
 ""
 "*
   return emit_asm_call(operands,0);
 "
)
```

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 3 13/25

Call Operation in spim3.md

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Call Operation in spim3.md

```
(define_insn "call_value"
 [(set (match_operand:SI 0 "register_operand" "=r")
       (call (match_operand:SI 1 "memory_operand" "m")
             (match_operand:SI 2 "immediate_operand" "i")))
  (clobber (reg:SI 31))
 ]
 ""
 "*
   return emit_asm_call(operands,1);
 "
)
```

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 3 14/25

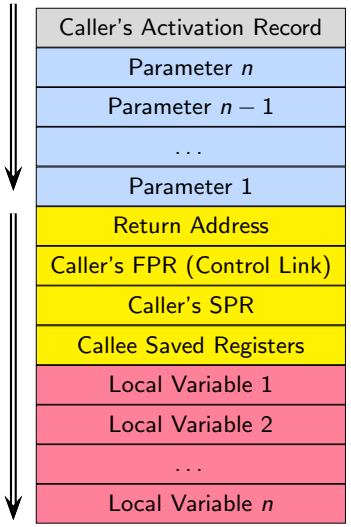
Call Operation in spim3.md

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Activation Record Generation during `Call`

- Operations performed by caller
  - Push parameters on stack.
  - Load return address in return address register.
  - Transfer control to Callee.
- Operations performed by callee
  - Push Return address stored by caller on stack.
  - Push caller's Frame Pointer Register.
  - Push caller's Stack Pointer.
  - Save callee saved registers, if used by callee.
  - Create local variables frame.
  - Start callee body execution.



Activation Record Generation during `Call`

Notes

Prologue in `spim3.md`

```
(define_expand "prologue"
 [(clobber (const_int 0))]
 ""
 {
    spim_prologue();
    DONE;
 })
```

```
(set (mem:SI (reg:SI $sp))
     (reg:SI 31 $ra))

(set (mem:SI (plus:SI (reg:SI $sp)
                (const_int -4 )))
     (reg:SI $sp))

(set (mem:SI (plus:SI (reg:SI $sp)
                (const_int -8 )))
     (reg:SI $fp))

(set (reg:SI $fp)
     (reg:SI $sp))

(set (reg:SI $sp)
       (plus:SI (reg:SI $fp)
           (const_int -36)))
```


Prologue in `spim3.md`

Notes

Epilogue in spim3.md

```
(define_expand "epilogue"
 [(clobber (const_int 0))]
 ""

    spim_epilogue();
    DONE;
 )
```

```
(set (reg:SI $sp)
     (reg:SI $fp))

(set (reg:SI $fp)
     (mem:SI (plus:SI (reg:SI $sp)
        (const_int -8 ))))

(set (reg:SI $ra)
(mem:SI (reg:SI $sp)))

(parallel [
    (return)
    (use (reg:SI $ra))])
```

Epilogue in spim3.md

Notes

Part 3

Constructs Supported in Level 4

Operations Required in Level 4

Operation	Primitive Variants	Implementation	Remark
$Src_1 < Src_2$? goto L : PC	$CC \leftarrow R_i < R_j$ CC < 0 ? goto L : PC	blt r_i, r_j,L	
$Src_1 > Src_2$? goto L : PC	$CC \leftarrow R_i > R_j$ CC > 0 ? goto L : PC	bgt r_i, r_j,L	
$Src_1 \leq Src_2$? goto L : PC	$CC \leftarrow R_i \leq R_j$ CC ≤ 0 ? goto L : PC	ble r_i, r_j,L	
$Src_1 \geq Src_2$? goto L : PC	$CC \leftarrow R_i \geq R_j$ CC ≥ 0 ? goto L : PC	bge r_i, r_j,L	

Operations Required in Level 4

Notes

Operations Required in Level 4

Operation	Primitive Variants	Implementation	Remark
$Src_1 == Src_2$? goto L : PC	$CC \leftarrow R_i == R_j$ CC $== 0$? goto L : PC	beq r_i, r_j,L	
$Src_1 \neq Src_2$? goto L : PC	$CC \leftarrow R_i \neq R_j$ CC $\neq 0$? goto L : PC	bne r_i, r_j,L	

Operations Required in Level 4

Notes

Conditional compare in spim4.md

```
(define_code_iterator cond_code
     [lt ltu eq ge geu gt gtu le leu ne])

(define_expand "cmpsi"
 [(set (cc0) (compare
             (match_operand:SI 0 "register_operand" "")
             (match_operand:SI 1 "nonmemory_operand" "")))]
 ""
 {
     compare_op0=operands[0];
     compare_op1=operands[1];
     DONE;
 }
)
```

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 4 20/25

Conditional compare in spim4.md

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 4 21/25

Branch pattern in spim4.md

```
(define_expand "b<code>"
 [(set (pc) (if_then_else (cond_code:SI (match_dup 1)
             (match_dup 2))
       (label_ref (match_operand 0 "" ""))
       (pc)))]
 ""
 {
    operands[1]=compare_op0;
    if(immediate_operand(compare_op1,SImode))
    {
       operands[2]=force_reg(SImode,compare_op1);
    }
    else
    {
       operands[2]=compare_op1;
    }
 }
)
```

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 4 21/25

Branch pattern in spim4.md

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Branch pattern in spim4.md

```
(define_insn "*insn_b<code>"
  [(set (pc)
        (if_then_else
   (cond_code:SI
      (match_operand:SI 1 "register_operand" "r")
              (match_operand:SI 2 "register_operand" "r"))
         (label_ref (match_operand 0 "" ""))
         (pc)))]
        ""
        "*
                return conditional_insn(<CODE>,operands,0);
        "
)
```


July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 4 22/25

Branch pattern in spim4.md

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Branch pattern in spim4.md

```
(define_insn "*insn_reverse_b<code>"
  [(set (pc)
        (if_then_else
   (cond_code:SI
      (match_operand:SI 1 "register_operand" "r")
              (match_operand:SI 2 "register_operand" "r"))
         (pc)
         (label_ref (match_operand 0 "" ""))))]
        ""
        "*
                return conditional_insn(<CODE>,operands,1);
        "
)
```


July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 4 23/25

Branch pattern in spim4.md

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Support for Branch pattern in spim4.c

```
char *
conditional_insn (enum rtx_code code,rtx operands[], int isRev)
{   if (! isRev)
    {    switch(code)
         {
              case EQ:return "beq %1, %2, %l0";
              case NE:return "bne %1, %2, %l0";
              case GE:return "bge %1, %2, %l0";
              case GT:return "bgt %1, %2, %l0";
              case LT:return "blt %1, %2, %l0";
              case LE:return "ble %1, %2, %l0";
              default: /* Error. Issue ICE */
         }
  }
  else
  {     /* Similar switch with operations reversed */
  }
}
```

July 09 Spim MD Levels 2,3,4: Constructs Supported in Level 4 24/25

Support for Branch pattern in spim4.c

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Lab Exercises for Spim Machine Descriptions Levels 2,3,4

Will be given in the lab :-)