GCC RTL Representation
(3) 각 RTX 의 표현 예제와 Machine Description 개요

정원교

2005년 9월 23일

목 차

제 1 절 18 주 문서를 시작하며

이번 강의에서는 RTL 을 구성하는 RTX 가 실제로 어떻게 사용되는지에 대해 예제로 알아본다. 이 사용처에 대한 참고서는 각 machine 을 위한 MD (Machine Description) 파일을 사용할 것이다.

제 2 절 RTX 의 표현 예제

우선 각 표현에 대한 예제 코드를 제시하기 전에 밝힐 것은 아래에 사용되는 예제는 거의 모두 GCC 용 $prefix/gcc/config/i386/i386.md 파일에서 추출한 것이며, 각 RTX 를 위한 설명은 이전 16 주 문서에서 이미 언급하였기 때문에 따로 설명하지 않는다. 설명이 필요하다면 이해를 돕기 위한 문장만 첨가할 것이다.

각 RTX 의 우측에 해당 표현식의 쓰임새를 요약적으로 표현한 형식이 함께 들어가 있음을 확인 할 수 있을 것이다.

ABS e 1 (abs:*m x*)

```
(abs:SF (match_operand:SF 1 "memory_operand" "0"))
(abs:SF (match_dup 1))
```

ADDR_DIFF_VEC eEee0 x (addr_diff_vec:m base [lr0 lr1 ...] min max flags)

```
(addr_diff_vec:DI (label_ref:SI 2389)
        [
            (label_ref:SI 1998)
            (label_ref:SI 2392)
            (label_ref:SI 2392)
            (label_ref:SI 2392)
            (label_ref:SI 2045)
            (label_ref:SI 2093)
            (label_ref:SI 2141)
            (label_ref:SI 2189)
            (label_ref:SI 2237)
            (label_ref:SI 2285)
            (label_ref:SI 2333)
        ]
        (const_int 0 [0x0])
        (const_int 0 [0x0]))
```

ADDR_VEC E x (addr_vec:*m [lr0 lr1 ...]*)

```
(addr_vec:SI[
    (label_ref:SI 148)
    (label_ref:SI 148)
    (label_ref:SI 148)
    (label_ref:SI 156)
    (label_ref:SI 156)
    (label_ref:SI 156)
    (label_ref:SI 156)
    (label_ref:SI 74)
    (label_ref:SI 146)
    (label_ref:SI 156)
    (label_ref:SI 156)
    (label_ref:SI 156)
    (label_ref:SI 156)
```

```
    (label_ref:SI 112)
    (label_ref:SI 129)
] )
```

ADDRESS e m

아직 제대로 된 예제를 찾지 못하였음.

ADDRESSOF eit o (addressof:*m reg*)

```
(addressof:DI (reg/v:SI 342) 341 0x206edfb0)
```

AND ee c (and:*m x y*)

```
(and (ge (minus (match_dup 0) (pc))
         (const_int -128))
     (lt (minus (match_dup 0) (pc))
         (const_int 124)))
(and (eq_attr "type" "call")
     (match_operand 1 "constant_call_address_operand" ""))
```

ASHIFT ee 2 (ashift:*m x c*)

```
(ashift:DI (match_operand:DI 1 "register_operand" "0")
           (const_int 32))
(ashift:DI (match_dup 1)
           (const_int 32))
(ashift:SI (match_operand:SI 1 "nonimmediate_operand" "")
           (match_operand:QI 2 "nonmemory_operand" ""))
```

ASHIFTRT ee 2 (ashiftrt:*m x c*)

```
(ashiftrt:SI (match_dup 1) (const_int 31))
(ashiftrt:DI (match_dup 4) (const_int 63))
(ashiftrt:DI (match_operand:DI 1 "shiftdi_operand" "")
             (match_operand:QI 2 "nonmemory_operand" ""))
```

ASM_INPUT s x (asm_input *s*)

```
(asm_input:SI ("1")
(asm_input:SI ("2")
(asm_input:SI ("b")
```

ASM_OPERANDS ssiEEsi x

```
(asm_operands/v ("movl $-1,%%eax
        xorl %%edx,%%edx
        repe; scasl
        je 1f
        xorl -4(%%edi),%%eax
        subl $4,%%edi
        bsfl %%eax,%%edx
1:      subl %%ebx,%%edi
        shll $3,%%edi
        addl %%edi,%%edx") ("=d") 0[
```

```
            (reg:SI 30)
            (reg/v:SI 22)
            (reg/v:SI 22)
        ]
        [
            (asm_input:SI ("1"))
            (asm_input:SI ("2"))
            (asm_input:SI ("b"))
        ]  ("1.c") 21)
```

ATTR_FLAG s x (attr_flag *name*)

```
(attr_flag "forward")
(attr_flag "backward")
```

ATTR s x (attr *name*)

```
(attr "modrm")
(attr "type")
```

BARRIER iuu x

```
(barrier 56 55 57)
(barrier 86 85 14)
```

CALL_INSN iuueieee i

```
(call_insn 18 16 20 (set (reg:SI 0 %eax)
        (call (mem:QI (symbol_ref:SI ("ggc_alloc")) 0)
            (const_int 16 [0x10]))) -1 (nil)
    (nil)
    (nil))
(call_insn 389 387 391 (call (mem:QI (symbol_ref:SI ("fancy_abort")) 0)
        (const_int 16 [0x10])) -1 (nil)
    (nil)
    (nil))
(call_insn 42 40 44 (set (reg:SI 0 %eax)
        (call (mem:QI (symbol_ref:SI ("memset")) 0)
            (const_int 16 [0x10]))) -1 (nil)
    (nil)
    (nil))
```

CALL_PLACEHOLDER uuuu x

```
(call_placeholder 669 663 0 0)
```

CALL ee x (call *function nargs*)

```
(call (match_operand:QI 0 "" "")
      (match_operand:SI 1 "" ""))
(call (mem:QI (match_operand:SI 0 "constant_call_address_operand" ""))
      (match_operand:SI 1 "" ""))
```

CC0 o

```
(cc0)
```

CLOBBER e x (clobber x)

```
(clobber (mem:BLK (scratch)))
(clobber (reg:CC 17))
(clobber (match_scratch:SI 3 "=&r"))
```

CODE_LABEL iuu00iss x

```
(code_label 61 60 0 9 "" [num uses: 0])
(code_label 106 105 108 75 "" [num uses: 0])
```

COMPARE ee 2 (compare:m x y)

```
(compare:CC (match_operand:DI 0 "nonimmediate_operand" "")
            (match_operand:DI 1 "x86_64_general_operand" ""))
(compare (and:SI (match_operand:SI 1 "nonimmediate_operand" "%0,0")
                 (match_operand:SI 2 "general_operand" "rim,ri"))
         (const_int 0))
(compare:CC (mem:BLK (match_dup 4))
            (mem:BLK (match_dup 5)))
```

CONCAT ee o

아직 제대로 된 예제를 찾지 못하였음.

COND_EXEC ee x (cond_exec *[cond expr]*)

```
(cond_exec (match_dup 7)
        (set (match_dup 0) (match_dup 4)))
(cond_exec (match_op_dup 1 [(match_dup 5) (const_int 0)])
        (set (match_dup 0) (match_dup 4)))
```

COND Ee x (cond *[test1 value1 test2 value2 ...] default*)

```
(cond [(eq_attr "type" "incdec,setcc,icmov,ibr,str,idiv,sse,mmx")
         (const_int 0)
       (eq_attr "i387" "1")
         (const_int 0)
       (eq_attr "type" "alu1,negnot,alu,icmp,imovx,ishift,imul,push,pop")
         (symbol_ref "ix86_attr_length_immediate_default(insn,1)")
       (eq_attr "type" "imov,test")
         (symbol_ref "ix86_attr_length_immediate_default(insn,0)")
       (eq_attr "type" "call")
         (if_then_else (match_operand 0 "constant_call_address_operand" "")
           (const_int 4)
           (const_int 0))
       (eq_attr "type" "callv")
         (if_then_else (match_operand 1 "constant_call_address_operand" "")
           (const_int 4)
           (const_int 0))
       (eq_attr "type" "ibr")
         (if_then_else (and (ge (minus (match_dup 0) (pc))
                                (const_int -128))
```

```
                        (lt (minus (match_dup 0) (pc))
                         (const_int 124)))
              (const_int 1)
              (const_int 4))
         ]
         (symbol_ref "abort(),1"))
(cond [(eq_attr "type" "other,multi,fistp")
         (const_int 16)
       ]
       (plus (plus (attr "modrm")
                   (plus (attr "prefix_0f")
                         (plus (attr "i387")
                               (const_int 1))))
             (plus (attr "prefix_rep")
                   (plus (attr "prefix_data16")
                         (plus (attr "length_immediate")
                               (attr "length_address"))))))
(cond [(eq_attr "type" "other,multi")
         (const_string "unknown")
       (and (eq_attr "type" "icmp,test,imov")
            (and (match_operand 0 "memory_displacement_operand" "")
                 (match_operand 1 "immediate_operand" "")))
         (const_string "true")
       (and (eq_attr "type" "alu,ishift,imul,idiv")
            (and (match_operand 0 "memory_displacement_operand" "")
                 (match_operand 2 "immediate_operand" "")))
         (const_string "true")
      ]
      (const_string "false"))
```

CONST_DOUBLE CONST_DOUBLE_FORMAT o

```
(const_double:SF 5.411961078643798828125e-1 [0x0.8a8bd4p+0])
(const_double (const_int 0 [0x0]) -1431655765 [0xaaaaaaab] 0 [0x0])
```

CONST_INT w o (const_int *i*)

```
(const_int 1)
(const_int 0)
(const_int 4)
```

CONST_STRING s o (const_string *str*)

```
(const_string "other")
(const_string "unknown")
(const_string "both")
```

CONST_VECTOR E x (const_vector:*m [x0 x1 ...]*)

```
(const_vector:V8QI [(const_int 1)
                    (const_int 1)
                    (const_int 1)
                    (const_int 1)
                    (const_int 1)
```

```
                    (const_int 1)
                    (const_int 1)
                    (const_int 1)])
(const_vector:V4HI [(const_int 1)
                    (const_int 1)
                    (const_int 1)
                    (const_int 1)])
(const_vector:V4SI [(const_int 32768)
                    (const_int 32768)
                    (const_int 32768)
                    (const_int 32768)])
```

CONST e o (const:m exp)

```
(const (symbol_ref "ix86_cpu"))
```

CONSTANT_P_RTX e x

아직 제대로 된 예제를 찾지 못하였음.

DEFINE_ASM_ATTRIBUTES V x (define_asm_attributes [*attr-sets*])

```
(define_asm_attributes
  [(set_attr "length" "128")
   (set_attr "type" "multi")])
```

DEFINE_ATTR sse x (define_attr *name list-of-values default*)

```
(define_attr "cpu" "i386,i486,pentium,pentiumpro,k6,athlon,pentium4"
  (const (symbol_ref "ix86_cpu")))
(define_attr "i387" ""
  (if_then_else (eq_attr "type" "fmov,fop,fop1,fsgn,fmul,fdiv")
    (const_int 1)
    (const_int 0)))
```

'sse' 문법에 따라 문자열+문자열+표현식 형태로 작성되어 있다.

DEFINE_COMBINE Ess x

아직 제대로 된 예제를 찾지 못하였음.

DEFINE_COND_EXEC Ess x

```
(define_cond_exec
  [(match_operator 0 "arm_comparison_operator"
    [(match_operand 1 "cc_register" "")
     (const_int 0)])]
  "TARGET_ARM"
  ""
)
(define_cond_exec
  [(match_operator 0 "predicate_operator"
     [(match_operand:BI 1 "register_operand" "c")
      (const_int 0)])]
  ""
  "(%J0)")
```

DEFINE_DELAY eE x

```
(define_delay (eq_attr "type" "call")
  [(eq_attr "in_call_delay" "true") (nil) (nil)])
(define_delay (eq_attr "type" "branch")
  [(eq_attr "in_branch_delay" "true")
   (nil) (eq_attr "in_annul_branch_delay" "true")])
```

DEFINE_EXPAND sEss x

```
(define_expand "cmpdi"
  [(set (reg:CC 17)
        (compare:CC (match_operand:DI 0 "nonimmediate_operand" "")
                    (match_operand:DI 1 "x86_64_general_operand" "")))]
  ""
{
  if (GET_CODE (operands[0]) == MEM && GET_CODE (operands[1]) == MEM)
    operands[0] = force_reg (DImode, operands[0]);
  ix86_compare_op0 = operands[0];
  ix86_compare_op1 = operands[1];
  DONE;
})
(define_expand "abstf2"
  [(parallel [(set (match_operand:TF 0 "nonimmediate_operand" "")
                   (neg:TF (match_operand:TF 1 "nonimmediate_operand" "")))
              (clobber (reg:CC 17))])]
  "TARGET_80387"
  "ix86_expand_unary_operator (ABS, TFmode, operands); DONE;")
(define_expand "blt"
  [(set (pc)
        (if_then_else (match_dup 1)
                      (label_ref (match_operand 0 "" ""))
                      (pc)))]
  ""
  "ix86_expand_branch (LT, operands[0]); DONE;")
```

DEFINE_FUNCTION_UNIT siieiiV x

```
(define_function_unit "pent_np" 1 0
  (and (eq_attr "cpu" "pentium")
       (eq_attr "type" "imul"))
  11 11)
(define_function_unit "ppro_p0" 1 0
  (and (eq_attr "cpu" "pentiumpro")
       (eq_attr "type" "fop,fop1,fsgn,fistp"))
  3 1)
(define_function_unit "ppro_p01" 2 0
  (and (and (eq_attr "cpu" "pentiumpro")
            (eq_attr "type" "imov,fmov"))
       (eq_attr "memory" "none"))
  1 1)
```

DEFINE_INSN_AND_SPLIT sEsTsESV x

```
(define_insn_and_split "*fix_truncdi_1"
  [(set (match_operand:DI 0 "nonimmediate_operand" "=m,?r")
        (fix:DI (match_operand 1 "register_operand" "f,f")))]
  "TARGET_80387 && FLOAT_MODE_P (GET_MODE (operands[1]))
   && !reload_completed && !reload_in_progress
   && (!SSE_FLOAT_MODE_P (GET_MODE (operands[1])) || !TARGET_64BIT)"
  "#"
  "&& 1"
  [(const_int 0)]
{
  operands[2] = assign_386_stack_local (HImode, 1);
  operands[3] = assign_386_stack_local (HImode, 2);
  if (memory_operand (operands[0], VOIDmode))
    emit_insn (gen_fix_truncdi_memory (operands[0], operands[1],
                                       operands[2], operands[3]));
  else
    {
      operands[4] = assign_386_stack_local (DImode, 0);
      emit_insn (gen_fix_truncdi_nomemory (operands[0], operands[1],
                                           operands[2], operands[3],
                                           operands[4]));
    }
  DONE;
}
  [(set_attr "type" "fistp")])
(define_insn_and_split "*pushv2sf"
  [(set (match_operand:V2SF 0 "push_operand" "=<")
        (match_operand:V2SF 1 "nonmemory_operand" "y"))]
  "TARGET_3DNOW"
  "#"
  ""
  [(set (reg:SI 7) (plus:SI (reg:SI 7) (const_int -8)))
   (set (mem:V2SF (reg:SI 7)) (match_dup 1))]
  ""
  [(set_attr "type" "mmx")])
(define_insn_and_split "*pushv8qi"
  [(set (match_operand:V8QI 0 "push_operand" "=<")
        (match_operand:V8QI 1 "nonmemory_operand" "y"))]
  "TARGET_MMX"
  "#"
  ""
  [(set (reg:SI 7) (plus:SI (reg:SI 7) (const_int -8)))
   (set (mem:V8QI (reg:SI 7)) (match_dup 1))]
  ""
  [(set_attr "type" "mmx")])
```

DEFINE_INSN sEsTV x

```
(define_insn "cmpdi_ccno_1_rex64"
  [(set (reg 17)
        (compare (match_operand:DI 0 "nonimmediate_operand" "r,?mr")
                 (match_operand:DI 1 "const0_operand" "n,n")))]
  "TARGET_64BIT && ix86_match_ccmode (insn, CCNOmode)"
```

```
  "@
   test{q}\t{%0, %0|%0, %0}
   cmp{q}\t{%1, %0|%0, %1}"
  [(set_attr "type" "test,icmp")
   (set_attr "length_immediate" "0,1")
   (set_attr "mode" "DI")])
(define_insn "*extendsidi2_1"
  [(set (match_operand:DI 0 "nonimmediate_operand" "=*A,r,?r,?*o")
        (sign_extend:DI (match_operand:SI 1 "register_operand" "0,0,r,r")))
   (clobber (reg:CC 17))
   (clobber (match_scratch:SI 2 "=X,X,X,&r"))]
  "!TARGET_64BIT"
  "#")
(define_insn "*addhi_1"
  [(set (match_operand:HI 0 "nonimmediate_operand" "=rm,r")
        (plus:HI (match_operand:HI 1 "nonimmediate_operand" "%0,0")
                 (match_operand:HI 2 "general_operand" "ri,rm")))
   (clobber (reg:CC 17))]
  "TARGET_PARTIAL_REG_STALL
   && ix86_binary_operator_ok (PLUS, HImode, operands)"
{
  switch (get_attr_type (insn))
    {
    case TYPE_INCDEC:
      if (operands[2] == const1_rtx)
        return "inc{w}\t%0";
      else if (operands[2] == constm1_rtx
               || (GET_CODE (operands[2]) == CONST_INT
                   && INTVAL (operands[2]) == 65535))
        return "dec{w}\t%0";
      abort();

    default:
      /* Make things pretty and `subl $4,%eax' rather than `addl $-4, %eax'.
         Exceptions: -128 encodes smaller than 128, so swap sign and op.  */
      if (GET_CODE (operands[2]) == CONST_INT
          && (INTVAL (operands[2]) == 128
              || (INTVAL (operands[2]) < 0
                  && INTVAL (operands[2]) != -128)))
        {
          operands[2] = GEN_INT (-INTVAL (operands[2]));
          return "sub{w}\t{%2, %0|%0, %2}";
        }
      return "add{w}\t{%2, %0|%0, %2}";
    }
}
  [(set (attr "type")
     (if_then_else (match_operand:HI 2 "incdec_operand" "")
        (const_string "incdec")
        (const_string "alu")))
   (set_attr "mode" "HI")])
```

DEFINE_PEEPHOLE EsTV x

```
(define_peephole
  [(parallel [(set (match_operand 0 "" "")
                   (call (mem:SI (match_operand:SI 1 "call_operand_address" "ps"))
                         (match_operand 2 "" "")))
              (clobber (reg:SI 15))])
   (set (pc) (label_ref (match_operand 3 "" "")))]
  "! TARGET_V9
   && short_branch (INSN_UID (insn), INSN_UID (operands[3]))
   && (USING_SJLJ_EXCEPTIONS || ! can_throw_internal (ins1))"
  "call\\t%a1, %2\\n\\tadd\\t%%o7, (%l3-.-4), %%o7")
(define_peephole
  [(set (match_operand:SI 0 "register_operand" "=t")
        (match_operand:SI 1 "register_operand" "d"))
   (set (pc)
        (if_then_else (match_operator:SI 2 "equality_op" [(match_dup 0)
                                                          (const_int 0)])
                      (match_operand 3 "pc_or_label_operand" "")
                      (match_operand 4 "pc_or_label_operand" "")))]
  "TARGET_MIPS16
   && GET_CODE (operands[0]) == REG
   && REGNO (operands[0]) == 24
   && dead_or_set_p (insn, operands[0])
   && GET_CODE (operands[1]) == REG
   && M16_REG_P (REGNO (operands[1]))"
  "*
{
  if (operands[3] != pc_rtx)
    return \"%*b%C2z\\t%1,%3\";
  else
    return \"%*b%N2z\\t%1,%4\";
}"
  [(set_attr "type"     "branch")
   (set_attr "mode"     "none")
   (set_attr "length"   "8")])
```

DEFINE_PEEPHOLE2 EsES x

```
(define_peephole2
  [(match_scratch:DI 2 "r")
   (set (match_operand:DI 0 "push_operand" "")
        (match_operand:DI 1 "immediate_operand" ""))]
  "TARGET_64BIT && !symbolic_operand (operands[1], DImode)
   && !x86_64_immediate_operand (operands[1], DImode)"
  [(set (match_dup 2) (match_dup 1))
   (set (match_dup 0) (match_dup 2))]
  "")
(define_peephole2
  [(set (match_operand 0 "register_operand" "")
        (const_int -1))]
  "(GET_MODE (operands[0]) == HImode
    || GET_MODE (operands[0]) == SImode
```

```
     || (GET_MODE (operands[0]) == DImode && TARGET_64BIT))
    && (optimize_size || TARGET_PENTIUM)
    && peep2_regno_dead_p (0, FLAGS_REG)"
   [(parallel [(set (match_dup 0) (const_int -1))
               (clobber (reg:CC 17))])]
   "operands[0] = gen_rtx_REG (GET_MODE (operands[0]) == DImode ? DImode : SImode,
                               true_regnum (operands[0]));")
(define_peephole2
   [(set (match_operand:DI 0 "push_operand" "")
         (match_operand:DI 1 "memory_operand" ""))
    (match_scratch:DI 2 "r")]
   "! optimize_size && ! TARGET_PUSH_MEMORY"
   [(set (match_dup 2) (match_dup 1))
    (set (match_dup 0) (match_dup 2))]
   "")
```

DEFINE_SPLIT EsES x

```
(define_split
   [(set (reg:CCFP 18)
         (compare:CCFP
           (match_operand:SF 0 "register_operand" "")
           (float (match_operand:SI 1 "register_operand" ""))))]
   "0 && TARGET_80387 && reload_completed"
   [(set (mem:SI (pre_dec:SI (reg:SI 7))) (match_dup 1))
    (set (reg:CCFP 18) (compare:CCFP (match_dup 0) (match_dup 2)))
    (parallel [(set (match_dup 1) (mem:SI (reg:SI 7)))
               (set (reg:SI 7) (plus:SI (reg:SI 7) (const_int 4)))])]
   "operands[2] = gen_rtx_MEM (Pmode, stack_pointer_rtx);
    operands[2] = gen_rtx_FLOAT (GET_MODE (operands[0]), operands[2]);")
(define_split
   [(set (match_operand:DF 0 "memory_operand" "")
         (neg:DF (match_operand:DF 1 "memory_operand" "")))
    (use (match_operand:DF 2 "" ""))
    (clobber (reg:CC 17))]
   ""
   [(parallel [(set (match_dup 0)
                    (neg:DF (match_dup 1)))
               (clobber (reg:CC 17))])])
(define_split
   [(set (match_operand 0 "register_operand" "")
         (match_operator 3 "binary_fp_operator"
            [(float (match_operand:SI 1 "register_operand" ""))
             (match_operand 2 "register_operand" "")]))]
   "TARGET_80387 && reload_completed
    && FLOAT_MODE_P (GET_MODE (operands[0]))"
   [(const_int 0)]
{
   operands[4] = ix86_force_to_memory (GET_MODE (operands[1]), operands[1]);
   operands[4] = gen_rtx_FLOAT (GET_MODE (operands[0]), operands[4]);
   emit_insn (gen_rtx_SET (VOIDmode, operands[0],
                           gen_rtx_fmt_ee (GET_CODE (operands[3]),
                                           GET_MODE (operands[3]),
```

```
                                              operands[4],
                                              operands[2])));
  ix86_free_from_memory (GET_MODE (operands[1]));
  DONE;
})
```

DIV ee 2 (div:*m x y*)

```
(div:QI (match_operand:HI 1 "register_operand" "0")
        (match_operand:QI 2 "nonimmediate_operand" "qm"))
(div:V4SF (match_operand:V4SF 1 "register_operand" "0")
          (match_operand:V4SF 2 "nonimmediate_operand" "xm"))
(div:DI (match_operand:DI 2 "register_operand" "1,0")
        (match_operand:DI 3 "nonimmediate_operand" "rm,rm"))
```

EQ_ATTR ss x (eq_attr *name value*)

```
(eq_attr "cpu" "pentium")
(eq_attr "type" "alu,alu1,ishift")
```

EQ ee < (eq:*m x y*)

```
(eq (symbol_ref "TARGET_PARTIAL_REG_STALL")
    (const_int 0))
(eq (match_operator 0 "ix86_comparison_operator"
                [(reg 17) (const_int 0)])
    (const_int 0))
```

EXPR_LIST ee x

```
(expr_list:REG_EQUAL (mult:SI (reg:SI 25)
            (const_int 4 [0x4]))
    (nil))
(expr_list:REG_EQUAL (symbol_ref:SI ("rtx_length"))
    (nil))
```

FFS e 1 (ffs:*m x*)

```
(ffs:SI (match_operand:SI 1 "general_operand" ""))
```

FIX e 1 (fix:*m x*)

```
(fix:DI (match_operand:XF 1 "register_operand" ""))
(fix:HI (match_operand 1 "register_operand" "f"))
(fix:DI (match_dup 1))
```

FLOAT_EXTEND e 1 (float_extend:*m x*)

```
(float_extend:DF (match_operand:SF 1 "nonimmediate_operand" "fY"))
(float_extend:DF (match_operand:SF 1 "register_operand" "0"))
```

FLOAT_TRUNCATE e 1 (float_truncate:*m x*)

```
(float_truncate:SF (match_operand:DF 1 "register_operand" ""))
(float_truncate:DF (match_dup 1))
```

FLOAT e 1 (float:*m x*)

```
(float (match_operand:SI 1 "nonimmediate_operand" "m,?r"))
(float:DF (match_operand:SI 1 "nonimmediate_operand" ""))
(float:SF (const_int 0))
```

GE ee < (ge:*m x y*)

```
(ge (minus (match_dup 0) (pc))
    (const_int -128))
(ge:QI (reg:CC 17) (const_int 0))
(ge:V2SI (match_operand:V2SF 1 "register_operand" "0")
         (match_operand:V2SF 2 "nonimmediate_operand" "ym"))
```

GEU ee < (geu:*m x y)*

```
(geu:QI (reg:CC 17) (const_int 0))
```

GT ee < (gt:*m x y*)

```
(gt:QI (reg:CC 17) (const_int 0))
(gt (match_operand:SF 1 "register_operand" "")
    (match_operand:SF 2 "nonimmediate_operand" ""))
```

GTU ee < (gtu:*m x y*)

```
(gtu:QI (reg:CC 17) (const_int 0))
```

HIGH e o (high:*m exp*)

```
(high:SF (match_operand:SF 1 "const_double_operand" "S"))
(high:SF (match_dup 1))
(high:SI (match_operand 2 "" ""))
```

IF_THEN_ELSE eee 3 (if_then_else *cond then else*)

```
(if_then_else (eq_attr "mode" "HI")
  (const_int 1)
  (const_int 0))
(if_then_else (match_operand 1 "memory_operand" "")
  (const_string "both")
  (const_string "store"))
(if_then_else (match_dup 1)
              (label_ref (match_operand 0 "" ""))
              (pc))
```

INCLUDE s x (include *pathname*)

```
(include "filestuff")
(include "BOGUS/filestuff")
```

INSN_LIST ue x

```
(insn_list 779 (insn_list 3495 (nil)))
```

INSN iuueiee i

```
(insn 11 9 12 (set (reg:SI 22)
        (plus:SI (mem/f:SI (reg:SI 17) 0)
            (const_int -1 [0xffffffff]))) -1 (nil)
    (nil))
(insn 239 238 241 (set (reg:SI 89)
        (ashift:SI (reg:SI 88)
            (const_int 2 [0x2]))) -1 (nil)
    (expr_list:REG_EQUAL (mult:SI (reg:SI 87)
            (const_int 4 [0x4]))
        (nil)))
```

IOR ee c (ior:*m x y*)

```
(ior (match_operand:SI 1 "register_operand" "")
     (match_operand:HI 1 "register_operand" ""))
(ior:DI (match_operand:DI 1 "nonimmediate_operand" "%0,0")
        (match_operand:DI 2 "x86_64_general_operand" "rem,re"))
(ior:SI (match_operand:SI 1 "nonimmediate_operand" "%0")
        (match_operand:SI 2 "general_operand" "rim"))
```

JUMP_INSN iuueiee0 i

```
(jump_insn 55 54 56 (set (pc)
        (label_ref 61)) -1 (nil)
    (nil))
(jump_insn 427 426 428 (set (pc)
        (label_ref 433)) -1 (nil)
    (nil))
```

LABEL_REF u00 o (label_ref *label*)

```
(label_ref (match_operand 0 "" ""))
(label_ref:DI (match_operand 3 "" ""))
(label_ref:DI (match_operand 3 "" "X"))
```

LE ee < (le:*m x y*)

```
(le:QI (reg:CC 17) (const_int 0))
```

LEU ee < (leu:*m x y*)

```
(leu:QI (reg:CC 17) (const_int 0))
```

LO_SUM ee o (lo_sum:*m x y*)

```
(lo_sum:SI (match_operand:SI 1 "register_operand" "r")
           (match_operand:SI 2 "immediate_operand" "in"))
(lo_sum:SI (match_operand:SI 1 "register_operand" "r")
        (unspec:SI [(match_operand:SI 2 "label_ref_operand" "")
                    (match_operand:SI 3 "" "")] 5))
```

LSHIFTRT ee 2 (lshiftrt:*m x c*)

```
(lshiftrt:SI (match_operand:SI 1 "register_operand" "Q")
             (const_int 8))
(lshiftrt:TI
  (mult:TI (zero_extend:TI
             (match_operand:DI 1 "nonimmediate_operand" ""))
           (zero_extend:TI
             (match_operand:DI 2 "register_operand" "")))
           (const_int 64))
```

LT ee < (lt:*m x y*)

```
(lt (minus (match_dup 0) (pc))
    (const_int 124))
(lt:QI (reg:CC 17) (const_int 0))
(lt (match_operand:DF 1 "register_operand" "")
    (match_operand:DF 2 "nonimmediate_operand" ""))
```

LTGT ee <

```
(ltgt:QI (reg:CC 17) (const_int 0))
```

LTU ee < (ltu:*m x y*)

```
(ltu:SI (reg:CC 17) (const_int 0))
(ltu:QI (reg:CC 17) (const_int 0))
(ltu (reg:CC 17) (const_int 0))
```

MATCH_DUP i m (match_dup *n*)

```
(match_dup 0)
(match_dup 1)
(match_dup 2)
```

MATCH_INSN is m (match_insn *predicate*)

아직 제대로 된 예제를 찾지 못하였음.

MATCH_OP_DUP iE m (match_op_dup:*m n[operands...]*)

```
(match_op_dup 1 [(match_dup 4) (const_int 0)])
(match_op_dup 1 [(match_dup 0) (match_dup 5)])
(match_op_dup 3 [(match_dup 1) (match_dup 2)])
```

MATCH_OPERAND iss m (match_operand:*m n predicate constraint*)

```
(match_operand 0 "constant_call_address_operand" "")
(match_operand:SI 1 "register_operand" "")
(match_operand 1 "immediate_operand" "")
```

MATCH_OPERATOR isE m (match_operator:*m n predicate [operands...]*)

```
(match_operator:QI 1 "ix86_comparison_operator"
  [(reg 17) (const_int 0)])
(match_operator:SF 3 "binary_fp_operator"
                [(match_operand:SF 1 "nonimmediate_operand" "%0")
                 (match_operand:SF 2 "nonimmediate_operand" "fm")])
(match_operator 1 "fcmov_comparison_operator"
       [(reg 17) (const_int 0)])
```

MATCH_PAR_DUP iE m (match_par_dup *n [subpat...]*)

```
(match_par_dup 3 [(set (match_operand:SI 0 "" "")
                       (match_operand:SI 1 "" ""))
                  (use (match_operand:SI 2 "" ""))])
(match_parallel 0 "load_multiple_operation"
                  [(set (match_operand:SI 1 "gpc_reg_operand" "=r")
                        (mem:SI (match_operand:SI 2 "gpc_reg_operand" "b")))])
```

MATCH_PARALLEL isE m (match_parallel *n predicate [subpat...]*)

```
(match_parallel 0 "load_multiple_operation"
                  [(set (match_operand:SI 1 "gpc_reg_operand" "=r")
                        (mem:SI (match_operand:SI 2 "gpc_reg_operand" "b")))])
(match_parallel 0 "store_multiple_operation"
                  [(set (mem:SI (match_operand:SI 1 "gpc_reg_operand" "b"))
                        (match_operand:SI 2 "gpc_reg_operand" "r"))
                   (clobber (match_scratch:SI 3 "X"))])
(match_parallel 0 "any_operand"
                  [(return)
                   (use (match_operand:SI 1 "register_operand" "l"))
                   (use (match_operand:SI 2 "call_operand" "s"))
                   (set (match_operand:DF 3 "gpc_reg_operand" "=f")
                        (match_operand:DF 4 "memory_operand" "m"))])
```

MATCH_SCRATCH is m (match_scratch:*m n constraint*)

```
(match_scratch:DI 2 "r")
(match_scratch:SI 2 "")
(match_scratch:SI 2 "=X,X,X,&r")
(match_scratch:DF 5 "=&1f,&1f")
```

MEM e0 o (mem:*m addr alias*)

```
(mem:SI (pre_dec:SI (reg:SI 7)))
(mem:SI (reg:SI 7))
(mem:BLK (scratch))
(mem:QI (match_operand:SI 0 "constant_call_address_operand" ""))
```

MINUS ee 2 (minus:*m x y*)

```
(minus (match_dup 0) (pc))
(minus:DI (match_operand:DI 0 "nonimmediate_operand" "rm,r")
          (match_operand:DI 1 "x86_64_general_operand" "re,mr"))
(minus:TF (match_operand:TF 1 "register_operand" "")
          (match_operand:TF 2 "register_operand" ""))
```

MOD ee 2 (mod:*m x y*)

```
(mod:DI (match_dup 1) (match_dup 2))
(mod:DI (match_dup 2) (match_dup 3))
(mod:SI (reg:SI 0) (match_dup 2))
```

MULT ee c (mult:*m x y*)

```
(mult (match_operand 1 "register_operand" "r")
      (match_operand 2 "const248_operand" "i"))
(mult:SI (match_operand:SI 1 "register_operand" "r")
         (match_operand:SI 2 "const248_operand" "n"))
(mult:DI (match_dup 1)
         (match_dup 2))
```

NE ee < (ne:*m x y*)

```
(ne (symbol_ref "flag_pic") (const_int 0))
(ne (symbol_ref "TARGET_MOVX")
    (const_int 0))
(ne (match_operator 0 "ix86_comparison_operator"
                [(reg 17) (const_int 0)])
    (const_int 0))
```

NEG e 1 (neg:*m x*)

```
(neg:DI (match_operand:DI 2 "x86_64_general_operand" "rme"))
(neg:SF (match_operand:SF 1 "nonimmediate_operand" "0,x#fr,0,0"))
(neg:SI (match_operand:SI 2 "general_operand" "rmni"))
```

NIL * x

```
(nil)
```

NOT e 1 (not:*m x*)

```
(not (match_operand 1 "memory_operand" ""))
(not:TI (match_operand:TI 1 "register_operand" "0"))
(not:V4SI
 (match_operator:V4SI 3 "sse_comparison_operator"
        [(match_operand:V4SF 1 "register_operand" "0")
         (match_operand:V4SF 2 "register_operand" "x")]))
```

NOTE iuu0ni x

```
(note 2 1 3 "" NOTE_INSN_DELETED)
(note 1 0 2 ("rtl.c") 312)
(note 6 5 7 0 NOTE_INSN_BLOCK_BEG)
```

ORDERED ee <

```
(ordered:QI (reg:CC 17) (const_int 0))
```

PARALLEL E x (parallel [*x0 x1 ...*])

```
(parallel [(set (match_dup 1) (mem:SI (reg:SI 7)))
           (set (reg:SI 7) (plus:SI (reg:SI 7) (const_int 4)))])
(parallel [(match_operand:QI 0 "" "=m")
           (match_operand:QI 1 "register_operand" "r")
           (match_operand:QI 2 "register_operand" "=&q")])
(parallel [(set (match_operand:HI 0 "register_operand" "")
                (mult:HI (match_operand:HI 1 "register_operand" "")
                         (match_operand:HI 2 "general_operand" "")))
           (clobber (reg:CC 17))])
```

PC o

```
(pc)
```

PHI E x

아직 제대로 된 예제를 찾지 못하였음.

PLUS ee c (plus:*m x y*)

```
(plus (plus (attr "modrm")
            (plus (attr "prefix_0f")
                  (plus (attr "i387")
                        (const_int 1))))
            (plus (attr "prefix_rep")
                  (plus (attr "prefix_data16")
                        (plus (attr "length_immediate")
                              (attr "length_address")))))
(plus:SI (reg:SI 7) (const_int 4))
(plus:SI (match_operand:SI 1 "nonimmediate_operand" "%0")
         (match_operand:SI 2 "general_operand" "rmni"))
```

POST_DEC e a (post_dec:*m x*)

```
(post_dec:QI (reg:QI 20))
(post_dec (reg:HI 32))
```

POST_INC e a (post_inc:*m x*)

```
(post_inc:QI (match_dup 0))
(post_inc:HI (match_operand:QI 1 "register_operand" ""))
```

POST_MODIFY ee a (post_modify:*m x y*)

```
(post_modify:SI (reg:SI 42) (plus (reg:SI 42)
                                  (reg:SI 48)))
```

PRE_DEC e a (pre_dec:*m x*)

```
(pre_dec:SI (reg:SI 7))
(pre_dec:DI (reg:DI 7))
```

PRE_INC e a (pre_inc:*m x*)

```
(pre_inc:QI (reg:QI 20))
(pre_inc:SI (match_dup 0))
```

PRE_MODIFY ee a (pre_modify:*m x expr*)

아직 제대로 된 예제를 찾지 못하였음.

PREFETCH eee x (prefetch:*m addr rw locality*)

```
(prefetch (match_operand:SI 0 "address_operand" "")
          (match_operand:SI 1 "const_int_operand" "")
          (match_operand:SI 2 "const_int_operand" ""))
(prefetch (match_operand:SI 0 "address_operand" "p")
          (const_int 0)
          (match_operand:SI 1 "const_int_operand" ""))
(prefetch (match_operand:SI 0 "address_operand" "p")
             (match_operand:SI 1 "const_int_operand" "n")
             (const_int 3))
```

QUEUED eeeee x

아직 제대로 된 예제를 찾지 못하였음.

RANGE_INFO uuEiiiiiibbii x

아직 제대로 된 예제를 찾지 못하였음.

RANGE_LIVE bi x

아직 제대로 된 예제를 찾지 못하였음.

RANGE_REG iiiiiiiitt x

아직 제대로 된 예제를 찾지 못하였음.

RANGE_VAR eti x

아직 제대로 된 예제를 찾지 못하였음.

REG i0 o (reg:*m n*)

```
(reg:CC 17)
(reg:SI 7)
(reg 17)
```

RESX i x

아직 제대로 된 예제를 찾지 못하였음.

RETURN x

```
(return)
```

ROTATE ee 2 (rotate:*m x c*)

```
(rotate:DI (match_operand:DI 1 "nonimmediate_operand" "")
           (match_operand:QI 2 "nonmemory_operand" ""))
(rotate:SI (match_operand:SI 1 "nonimmediate_operand" "")
           (match_operand:QI 2 "nonmemory_operand" ""))
(rotate:HI (match_operand:HI 1 "nonimmediate_operand" "")
           (match_operand:QI 2 "nonmemory_operand" ""))
```

ROTATERT ee 2 (rotatert:*m x c*)

```
(rotatert:DI (match_operand:DI 1 "nonimmediate_operand" "")
             (match_operand:QI 2 "nonmemory_operand" ""))
(rotatert:SI (match_operand:SI 1 "nonimmediate_operand" "")
             (match_operand:QI 2 "nonmemory_operand" ""))
(rotatert:HI (match_operand:HI 1 "nonimmediate_operand" "")
             (match_operand:QI 2 "nonmemory_operand" ""))
```

SCRATCH 0 o (scratch:*m*)

```
(scratch)
```

SEQUENCE E x (sequence [*insns* ...])

아직 제대로 된 예제를 찾지 못하였음.

SET_ATTR_ALTERNATIVE sE x (set_attr_alternative *name* [*value1 value2* ...])

```
(set_attr_alternative "length"
             [(if_then_else (match_operand:VOID 2 "m16_simm8_1" "")
                            (const_int 4)
                            (const_int 8))
              (if_then_else (match_operand:VOID 2 "m16_simm4_1" "")
                            (const_int 4)
                            (const_int 8))
              (const_int 4)])
(set_attr_alternative "length"
             [(const_int 4)
              (if_then_else (match_operand:VOID 2 "m16_uimm8_1" "")
                            (const_int 4)
                            (const_int 8))
              (const_int 4)])
```

SET_ATTR ss x (set_attr *name value-string*)

```
(set_attr "type" "test,icmp")
(set_attr "length_immediate" "0,1")
```

SET ee x (set *lval x*)

```
(set (reg:CC 17)
     (compare:CC (match_operand:DI 0 "nonimmediate_operand" "")
                 (match_operand:DI 1 "x86_64_general_operand" "")))
(set (reg:CCFP 18)
     (compare:CCFP
       (match_operand:XF 0 "register_operand" "f")
       (match_operand:XF 1 "register_operand" "f")))
(set (match_operand:HI 0 "register_operand" "=a")
     (unspec:HI [(reg 18)] 9))
```

SIGN_EXTEND e 1 (sign_extend:*m x*)

```
(sign_extend:DI (match_operand:SI 1 "register_operand" ""))
(sign_extend:DI (match_operand:SI 1 "register_operand" "0,0,r,r"))
(sign_extend:V2SI
   (ss_truncate:V2HI
      (fix:V2SI (match_operand:V2SF 1 "nonimmediate_operand" "ym"))))
```

SIGN_EXTRACT eee b (sign_extract:*m loc size pos*)

```
(sign_extract:SI (match_operand 1 "ext_register_operand" "Q")
                 (const_int 8)
                 (const_int 8))
(sign_extract:SI (match_operand:SI 1 "register_operand" "")
                 (match_operand:SI 2 "immediate_operand" "")
                 (match_operand:SI 3 "immediate_operand" ""))
(sign_extract:QI (match_operand 1 "ext_register_operand" "Q,Q")
                 (const_int 8)
                 (const_int 8))
```

SMAX ee c (smax:*m x y*)

```
(smax:V4SF (match_operand:V4SF 1 "register_operand" "0")
           (match_operand:V4SF 2 "nonimmediate_operand" "xm"))
(smax:V4HI (match_operand:V4HI 1 "register_operand" "0")
           (match_operand:V4HI 2 "nonimmediate_operand" "ym"))
(smax:V2SF (match_operand:V2SF 1 "register_operand" "0")
           (match_operand:V2SF 2 "nonimmediate_operand" "ym"))
```

SMIN ee c (smin:*m x y*)

```
(smin:V4SF (match_operand:V4SF 1 "register_operand" "0")
           (match_operand:V4SF 2 "nonimmediate_operand" "xm"))
(smin:V4HI (match_operand:V4HI 1 "register_operand" "0")
           (match_operand:V4HI 2 "nonimmediate_operand" "ym"))
(smin:V2SF (match_operand:V2SF 1 "register_operand" "0")
           (match_operand:V2SF 2 "nonimmediate_operand" "ym"))
```

SQRT e 1 (sqrt:*m x*)

```
(sqrt:SF (match_operand:SF 1 "nonimmediate_operand" ""))
(sqrt:SF (match_operand:SF 1 "nonimmediate_operand" "0#x,xm#f"))
(sqrt:DF (match_operand:DF 1 "nonimmediate_operand" "Ym"))
```

SS_MINUS ee 2 (ss_minus:*m x y*)

```
(ss_minus:V8QI (match_operand:V8QI 1 "register_operand" "0")
               (match_operand:V8QI 2 "nonimmediate_operand" "ym"))
(ss_minus:V4HI (match_operand:V4HI 1 "register_operand" "0")
                       (match_operand:V4HI 2 "nonimmediate_operand" "ym"))
```

SS_PLUS ee c (ss_plus:*m x y*)

```
(ss_plus:V8QI (match_operand:V8QI 1 "register_operand" "0")
              (match_operand:V8QI 2 "nonimmediate_operand" "ym"))
(ss_plus:V4HI (match_operand:V4HI 1 "register_operand" "0")
              (match_operand:V4HI 2 "nonimmediate_operand" "ym"))
```

SS_TRUNCATE e 1 (ss_truncate:*m x*)

```
(ss_truncate:V4QI (match_operand:V4HI 1 "register_operand" "0"))
(ss_truncate:V2HI (match_operand:V2SI 2 "register_operand" "y"))
(ss_truncate:V2HI
   (fix:V2SI (match_operand:V2SF 1 "nonimmediate_operand" "ym")))
```

STRICT_LOW_PART e x (strict_low_part *(subreg:m (reg:n r) 0)*)

```
(strict_low_part (match_operand:HI 0 "nonimmediate_operand" ""))
(strict_low_part (match_operand:HI 0 "nonimmediate_operand" "+rm,r"))
(strict_low_part (match_operand:QI 0 "nonimmediate_operand" "+qm,q"))
(strict_low_part (match_dup 1))
```

SUBREG ei x (subreg:*m reg bytenum*)

```
(subreg:QI
  (zero_extract:SI
    (match_operand 1 "ext_register_operand" "Q")
    (const_int 8)
    (const_int 8)) 0)
(subreg:TI (match_dup 2) 0)
(subreg:TI (match_operand:DF 0 "register_operand" "=Y") 0)
```

SYMBOL_REF s o (symbol_ref:*mode symbol*)

```
(symbol_ref "ix86_cpu")
(symbol_ref "ix86_attr_length_immediate_default(insn,1)")
(symbol_ref "/* Update immediate_length and other attributes! */ abort(),1")
```

TRAP_IF ee x

```
(trap_if (const_int 1) (const_int 5))
(trap_if (match_operator 0 "comparison_operator"
          [(match_dup 2) (const_int 0)])
         (match_operand 1 "const_int_operand" ""))
(trap_if (match_operator 0 "comparison_operator"
          [(reg 17) (const_int 0)])
         (match_operand 1 "const_int_operand" ""))
```

TRUNCATE e 1 (truncate:*m x)*

```
(truncate:DI
  (lshiftrt:TI
    (mult:TI (zero_extend:TI
               (match_operand:DI 1 "nonimmediate_operand" ""))
             (zero_extend:TI
               (match_operand:DI 2 "register_operand" "")))
    (const_int 64)))
(truncate:SI
  (lshiftrt:DI
    (mult:DI (zero_extend:DI
               (match_operand:SI 1 "nonimmediate_operand" ""))
             (zero_extend:DI
               (match_operand:SI 2 "register_operand" "")))
    (const_int 32)))
(truncate:HI (match_operand:SI 2 "nonimmediate_operand" "rm"))
(truncate:V4HI
  (lshiftrt:V4SI
    (mult:V4SI (sign_extend:V4SI
                (match_operand:V4HI 1 "register_operand" "0"))
```

```
                (sign_extend:V4SI
                 (match_operand:V4HI 2 "nonimmediate_operand" "ym")))
        (const_int 16)))
```

UDIV ee 2 (udiv:m x y)

```
(udiv:QI (match_operand:HI 1 "register_operand" "0")
         (match_operand:QI 2 "nonimmediate_operand" "qm"))
(udiv:DI (match_dup 1) (match_dup 2))
(udiv:SI (match_operand:SI 1 "register_operand" "0")
         (match_operand:SI 2 "nonimmediate_operand" "rm"))
```

UMAX ee c (umax:m x y)

```
(umax:V8QI (match_operand:V8QI 1 "register_operand" "0")
           (match_operand:V8QI 2 "nonimmediate_operand" "ym"))
```

UMIN ee c (umin:m x y)

```
(umin:V8QI (match_operand:V8QI 1 "register_operand" "0")
           (match_operand:V8QI 2 "nonimmediate_operand" "ym"))
```

UMOD ee 2 (umod:m x y)

```
(umod:DI (match_dup 1) (match_dup 2))
(umod:SI (match_dup 1) (match_dup 2))
(umod:HI (match_dup 1) (match_dup 2))
```

UNEQ ee <

```
(uneq:QI (reg:CC 17) (const_int 0))
```

UNGE ee <

```
(unge:QI (reg:CC 17) (const_int 0))
```

UNGT ee <

```
(ungt:QI (reg:CC 17) (const_int 0))
```

UNKNOWN * x

아직 제대로 된 예제를 찾지 못하였음.

UNLE ee <

```
(unle:QI (reg:CC 17) (const_int 0))
```

UNLT ee <

```
(unlt:QI (reg:CC 17) (const_int 0))
```

UNORDERED ee <

```
(unordered:QI (reg:CC 17) (const_int 0))
```

UNSIGNED_FIX e 1 (unsigned_fix:*m x*)

```
(unsigned_fix:SI (match_operand:TF 1 "general_operand" ""))
(unsigned_fix:DI (match_operand:TF 1 "general_operand" ""))
```

UNSIGNED_FLOAT e 1 (unsigned_float:*m x*)

```
(unsigned_float:TF (match_operand:SI 1 "register_operand" ""))
(unsigned_float:DF (match_operand:SI 1 "register_operand" "f"))
```

UNSPEC_VOLATILE Ei x (unspec_volatile [*operands ...*] *index*)

```
(unspec_volatile [(const_int 0)] 0)
(unspec_volatile:SI
 [(plus:SI (match_dup 0)
           (plus:SI (match_operand:SI 1 "symbolic_operand" "")
                    (minus:SI (pc) (match_operand 2 "" ""))))] 1)
(unspec_volatile:SI [(plus:SI (pc) (match_operand 1 "" ""))] 2)
```

UNSPEC Ei x (unspec [*operands ...*] *index*)

```
(unspec:HI
  [(compare:CCFP (match_operand 1 "register_operand" "f")
                 (match_operand 2 "const0_operand" "X"))] 9)
(unspec:TF [(match_operand:TF 1 "register_operand" "0")] 2)
(unspec:BLK [(reg:DI 21)
             (reg:DI 22)
             (reg:DI 23)
             (reg:DI 24)
             (reg:DI 25)
             (reg:DI 26)
             (reg:DI 27)
             (reg:DI 28)] 13)
```

US_MINUS ee 2 (us_minus:*m x y*)

```
(us_minus:V8QI (match_operand:V8QI 1 "register_operand" "0")
               (match_operand:V8QI 2 "nonimmediate_operand" "ym"))
(us_minus:V4HI (match_operand:V4HI 1 "register_operand" "0")
               (match_operand:V4HI 2 "nonimmediate_operand" "ym"))
```

US_PLUS ee c (us_plus:*m x y*)

```
(us_plus:V8QI (match_operand:V8QI 1 "register_operand" "0")
              (match_operand:V8QI 2 "nonimmediate_operand" "ym"))
(us_plus:V4HI (match_operand:V4HI 1 "register_operand" "0")
              (match_operand:V4HI 2 "nonimmediate_operand" "ym"))
```

US_TRUNCATE e 1 (us_truncate:*m x*)

```
(us_truncate:V4QI (match_operand:V4HI 1 "register_operand" "0"))
(us_truncate:V4QI (match_operand:V4HI 2 "register_operand" "y"))
```

USE e x (use x)

```
(use (match_operand:HI 2 "memory_operand" "m,m"))
(use (match_dup 2))
(use (match_operand:HI 2 "memory_operand" ""))
(use (match_operand:DF 2 "general_operand" "Y,0,*g#Yr,*rm"))
(use (reg:SI 19))
```

VALUE 0 o

아직 제대로 된 예제를 찾지 못하였음.

VEC_CONCAT ee x (vec_concat:*m vec1 vec2*)

```
(vec_concat:V8QI
 (ss_truncate:V4QI (match_operand:V4HI 1 "register_operand" "0"))
 (ss_truncate:V4QI (match_operand:V4HI 2 "register_operand" "y")))
(vec_concat:V4HI
 (ss_truncate:V2HI (match_operand:V2SI 1 "register_operand" "0"))
 (ss_truncate:V2HI (match_operand:V2SI 2 "register_operand" "y")))
(vec_concat:V2SF
   (plus:SF
      (vec_select:SF (match_operand:V2SF 1 "register_operand" "0")
                     (parallel [(const_int  0)]))
      (vec_select:SF (match_dup 1)
                     (parallel [(const_int 1)])))
   (plus:SF
      (vec_select:SF (match_operand:V2SF 2 "nonimmediate_operand" "y")
                     (parallel [(const_int  0)]))
      (vec_select:SF (match_dup 2)
                     (parallel [(const_int 1)]))))
```

VEC_DUPLICATE e x (vec_duplicate:*m vec*)

```
(vec_duplicate:V4SF (float:SF (const_int 0)))
(vec_duplicate:V4SF
  (float:V2SF (match_operand:V2SI 2 "nonimmediate_operand" "ym")))
(vec_duplicate:V4HI
  (truncate:HI (match_operand:SI 2 "nonimmediate_operand" "rm")))
(vec_duplicate:V4SF (float:SF (const_int 0)))
```

VEC_MERGE eee x (vec_merge:*m vec1 vec2 items*)

```
(vec_merge:V4SF
 (match_operand:V4SF 1 "register_operand" "0")
 (vec_select:V4SF (match_operand:V4SF 2 "register_operand" "x")
                  (parallel [(const_int 2)
                             (const_int 3)
                             (const_int 0)
                             (const_int 1)]))
 (const_int 3))
(vec_merge:V4SI
 (match_operator:V4SI 3 "sse_comparison_operator"
        [(match_operand:V4SF 1 "register_operand" "0")
         (match_operand:V4SF 2 "register_operand" "x")])
 (match_dup 1)
```

```
 (const_int 1))
(vec_merge:V4SF
 (vec_select:V4SF (match_operand:V4SF 1 "register_operand" "0")
                  (parallel [(const_int 2)
                             (const_int 0)
                             (const_int 3)
                             (const_int 1)]))
 (vec_select:V4SF (match_operand:V4SF 2 "register_operand" "x")
                  (parallel [(const_int 0)
                             (const_int 2)
                             (const_int 1)
                             (const_int 3)]))
 (const_int 5))
```

VEC_SELECT ee x (vec_select:*m vec1 selection*)

```
(vec_select:V4SF (match_operand:V4SF 2 "register_operand" "x")
                 (parallel [(const_int 2)
                            (const_int 3)
                            (const_int 0)
                            (const_int 1)]))
(vec_select:SF
 (match_operand:V4SF 1 "register_operand" "x")
 (parallel [(const_int 0)]))
(vec_select:V2SI (match_operand:V2SI 1 "register_operand" "0")
                 (parallel [(const_int 1)
                            (const_int 0)]))
```

XOR ee c (xor:*m x y*)

```
(xor:SI
  (zero_extract:SI (match_dup 0)
                   (const_int 8)
                   (const_int 8))
  (zero_extract:SI (match_dup 0)
                   (const_int 8)
                   (const_int 8)))
(xor:DI (match_operand:DI 1 "nonimmediate_operand" "")
        (match_operand:DI 2 "x86_64_general_operand" ""))
(xor:SI (match_dup 1) (const_int -1))
```

ZERO_EXTEND e 1 (zero_extend:*m x*)

```
(zero_extend:SI (match_operand:HI 1 "nonimmediate_operand" ""))
(zero_extend:TI
  (match_operand:DI 1 "nonimmediate_operand" ""))
(zero_extend:HI (match_operand:QI 1 "nonimmediate_operand" "0,qm"))
```

ZERO_EXTRACT eee b (zero_extract:*m loc size pos*)

```
(zero_extract:SI
   (match_operand 1 "ext_register_operand" "Q")
   (const_int 8)
   (const_int 8))
```

```
(zero_extract:SI (match_operand 0 "ext_register_operand" "")
                 (match_operand:SI 1 "immediate_operand" "")
                 (match_operand:SI 2 "immediate_operand" ""))
(zero_extract:SI
  (match_dup 0)
  (const_int 8)
  (const_int 8))
```

지금까지 각 RTX 의 쓰임새에 대한 예제를 언급하였다. 비록 각 RTX 의 자세한 설명은 달지 않았지만, 앞에서 작성했던 문서에서 그에 대한 기능 및 쓰임새를 충분히 알 수 있으리라 생각한다.

제 3 절 마치며

이번 주에는 Machine Description 에 대해서 개요를 적을려고 하였으나, 아직 때가 아닌 것 같아 적지 않습니다. 좀 더 GCC 컴파일러의 전반부를 살펴본 후 천천히 보도록 하겠습니다. 다음 주 문서부터 이에 대해 좀 더 심도 있게 접근해 보도록 하겠습니다.