GNU コンパイラ移植による独自アーキテクチャ用コンパイラ

gcc の SNKプロセッサへの移植のための記述

gccの移植はテスト用のアーキテクチャ名を local として ソース内の
gcc/config/local ディレクトリ中にアーキテクチャ固有の情報を記述する

■ gcc/config/local/local.h local.c

local.h はターゲットマクロの記述を行うヘッダファイルである
複雑なマクロ記述に関してはCの関数名を指定して local.c にその関数を記述する

■ gcc/config/local/t-local

gccインストール時の configureを行うときに組み込まれるMakeFileの断片である
ファイルが存在しない場合は指定されたターゲットやホストに
関しては追加すべき情報が無いことになる
SNKでは乗算命令等をライブラリとして組み込むためライブラリ libgcc1 の指定を行う

■ gcc/config/local/xm-local.h

コンパイラが実行されるマシンおよび
システムに関するマクロ定義を行うコンフィグレーションファイル.
ビッグエンディアンであるかどうか，int,char,short,long,longlongbit型が
何bitで構成されているか等の情報を記述する

■ gcc/config/local/local.md

マシンディスクリプション記述
以下のように命令動作パターンとアセンブラ出力の対応の記述をを行う

● 加算命令

中間言語RTLによる記述でGCCにおける標準命令をSNKの命令として表す

```
(define_insn "snk_add"
    [(set (match_operand:SI 0 "register_operand" "=r")
        (plus:SI (match_operand:SI 1 "register_operand" "%r")
                 (mach_operand:SI 2 "register_operand" "r")))]
    ""
    "ADD %0, %1, %2"
)


(define_expand "andsi3"
    [(set (match_operand:SI 0 "register_operand" "=r")
          (and:SI (match_operand:SI 1 "register_operand" "%r")
                  (match_operand:SI 2 "register_operand" "r")))]
    ""
    "{
       emit_insn (gen_snk_and (operands[0], operands[1], operand[2]));
       DONE;
    }"
)
```

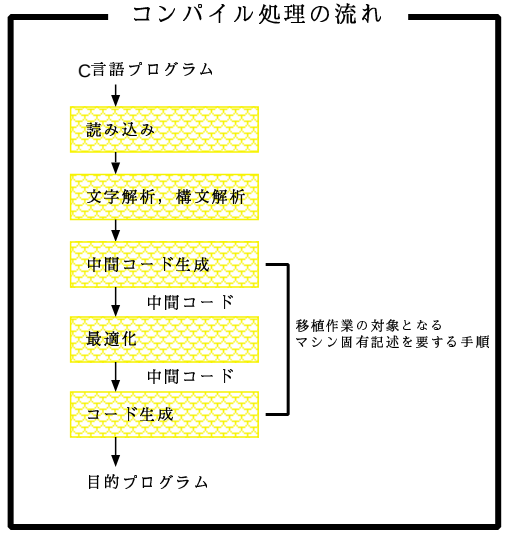
● 乗算命令

乗算，除算命令等はアセンブラ記述によりエミュレートする

```
LDA     3,0,1
LDA     4,0,2
SLT     1,1,0
SLT     2,2,0
ADD     2,1,2
LDA     5,1,0
AND     2,2,5
L1:
SLT     5,3,0
BZ      5,L2,0
NOT     3,3
LDA     3,1,3
L2:
SLT     5,4,0
BZ      5,L3,0
NOT     4,4
LDA     4,1,4
```

```
L3:
LDA     6,30,0
L4:
LDA     5,1,0
AND     5,4,5
BZ      5,L5,0
ADD     1,1,3
L5:
ADD     3,3,3
SR      4,4,0
LDA     6,-1,6
LDA     5,0,0
SLT     5,6,5
BZ      5,L4,0
BZ      2,L6,0
NOT     1,1
LDA     1,1,1
L6:
```

コンパイル処理の流れ



SNKプロセッサ

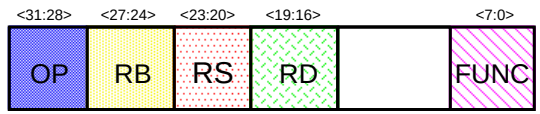
■ シンプルなノンパイプライン
32bit RISC型プロセッサ

■ 必要最小限の18命令を実装
乗算，除算命令は非実装のため
コンパイラレベルでの
エミュレーションを行う

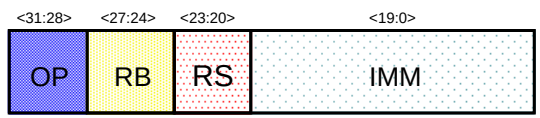
■ レジスタセット
プログラムカウンタ(PC)
命令レジスタ(op)
割り込みレジスタ(ir)
汎用レジスタ(gr0-7)

■ R形式，I形式の二つの
命令形式を持つ

R-Type instruction (RD <- RB func RS)
OP: ADD,AND,SLTU,SLT,NOT,SEQ,SR



I-Type instruction (RB <- RS op IMM)
OP: RBL,RBR,SB,LD,ST,LDA,RI,IN,OUT,BZ,BAL



OP	: Operation
FUNC	: R-Type function
RD	: Distination Register
RB	: Base Register
RS	: Source Register
IMM	: Immidiate

現状では binutils の SNK への移植が完了していないためライブラリが不足している
そのため gcc の make はエラーとなり完了することが出来ないが，仮のCコンパイラである xgcc でのアセンブラ出力によってマシン記述の検証を行っている