Vectorization or SIMDization

Vectorization or SIMDization (here-in-after referred as Vectorization only) is the process of making a program to use the SIMD instructions of a processor.

SIMD

SIMD expands to **S**ingle **I**nstruction **M**ultiple **D**ata. Processor that supports SIMD instructions would be capable of performing a single operation on multiple data (i.e. vector data).

For example, for the following program:

```
int a[4] = {1,2,3,4};  int b[4] = {4,3,2,1};
void vectorAdd ()  {      for(i=0; i<4; i++)      {        a[i] = a[i] + b[i];      }  }
```

In the assembly for a RISC architecture (without SIMD support), there would be four add operations. The assembly generated by GCC Ver.4.4.0 for 'frv' architecture is as follows:

```
vectorAdd:      sethi.p #hi(a), gr4      sethi #hi(b), gr5      setlo.p #lo(a), gr4      setlo #lo(b), gr5
ldi.p @(gr5,12), gr11      ld @(gr4,gr0), gr7      ldi.p @(gr4,4), gr6      ldi @(gr4,8), gr9      ldi.p
@(gr4,12), gr8      ld @(gr5,gr0), gr12      ldi.p @(gr5,4), gr10      ldi @(gr5,8), gr5      setlos #4,
gr13      sti gr13, @(gr16,#gprel12(i))      add.p gr12,gr7,gr7      add gr11,gr8,gr8      add.p
gr10,gr6,gr6      add gr5,gr9,gr5      sti gr8, @(gr4,12)      st gr7, @(gr4,gr0)      sti gr6, @(gr4,4)
sti.p gr5, @(gr4,8)      ret
```

But the assembly for a RISC architecture that supports SIMD for vectors of size '4' would have a single add operation on multiple data. The assembly would be as follows:

```
vectorAdd:      MOV  _a, R0;      // load 4 elements of 'a'. 'A' is a vector      V MOV  X(R0)+1, A
// register that can hold 4 elements      MOV  _b, R1;      // load 4 elements of 'a'. 'B' is a
vector      V MOV  X(R1)+1, B   // register that can hold 4 elements      V ADD  B, A      // single
add operation on multiple data      V MOV  A, X(R0)+1      rts
```

GCC

GCC performs loop-based vectorization. Vectorization support (loop) in GCC can be done by using either of the following two ways:

1. intrinsic support provided by GCC
2. autovectorization done by GCC

1. intrinsic support provided by GCC

Here we require to have an extension to the programming language ('C'). This extension is to provide the necessary vector data types. This extension can be done using typedef

For example,

```
typedef int v4si __attribute__ ((mode(V4SI)));
```

Here the data type 'v4si' is defined. And this vector type can hold '4' operands of type 'signed int'. The properties of new data type 'v4si' are defined by __attribute__ qualifier.

The input to GCC would be as follows:

```
typedef int v4si __attribute__ ((mode(V4SI)));
v4si a, b, c;
void vAddVectors ()  {      c = __builtin_addv4si (a, b);  }
```

The machine descriptions needs to updated for providing the information on the built-in functions.

2. autovectorization done by GCC

GCC automatically performs vectorization. It transforms scalar code into vector code. I.e. It transforms the operations on individual array elements one after the other into vector operations. This transformation done by GCC is called auto-vectorization. And this is done in loop optimization phase of SSA optimizations.

The machine descriptions needs to updated for providing the information on the SIMD instructions.