Computer Architecture 2023

Homework 4 Compile high level language into assembly (floating-point arithmetic)

The aim is to study how RISC-V arithmetic instructions are used to write a non-trivial program.

I will use Matrix mulitply program written in C as an example. Compile it into RISC-V 32 bit assembly language. Starting from a pseudo code for Matrix multiply. I ask you to do a floating-point version.

```
---------- Pseudo code ----------

n is the rank of matrix (n x n)
A, B, C are square matrix of rank n

matmul(n, A, B, C){

      for( i = 0; i < n; i++)
            for( j = 0; j < n; j++){
                  cij = C[i][j]
                  for( k = 0; k < n; k++)
                        cij += A[i][k] * B[k][j];
                  C[i][j] = cij;
            }
}
```

The first version of this pseudo code is implemented with an integer data. I transform that into a concrete C program. I use one-dimension integer array to store matrix. So A[i][j] is indexed by A[i+j*n].

```
--------   C program -------------
#include <stdio.h>

void minit(int n, int m, int* D){
      int i, j;
      for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                  D[i+j*n] = m;
}

void mprint(int n, int* D){
      int i, j;
```

```
      for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                  printf("%d ", D[i+j*n]);
      printf("\n");
}

void matmul(int n, int* A, int* B, int* C){
      int i, j, k;
      int cij;

      for( i = 0; i < n; i++)
            for( j = 0; j < n; j++){
                  cij = C[i+j*n];
                  for( k = 0; k < n; k++)
                        cij += A[i+k*n] * B[k+j*n];  /*   ****** */
                  C[i+j*n] = cij;
            }
}

int main(void){
      int A[9], B[9], C[9];
      int n;

      n = 3;
      minit(n,10,A);
      mprint(n,A);
      minit(n,20,B);
      mprint(n,B);
      minit(n,0,C);
      mprint(n,C);
      matmul(n,A,B,C);
      mprint(n,C);
}
```

I also include initialise of matrix, and matrix print so we can check that the program works correctly.

Here is the output from compile and run the above C program.

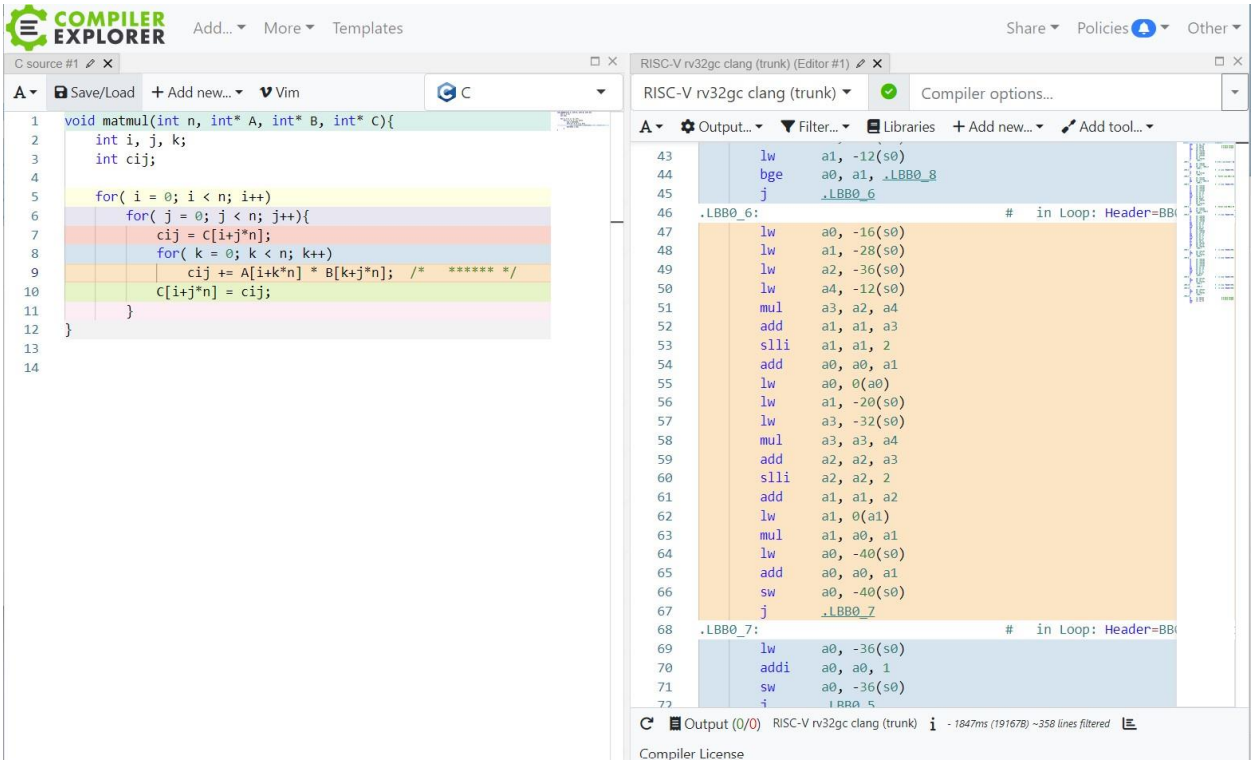
```
---- integer data --------------

C:\Users\prabhas\Dropbox\comparch-2023\matmul\lcc>mm
10 10 10 10 10 10 10 10 10
20 20 20 20 20 20 20 20 20
0 0 0 0 0 0 0 0 0
600 600 600 600 600 600 600 600 600
```

The inner product (the line labelled *****) is the result of 10*20 + 10*20 + 10*20 = 600

So, the program works correctly.

We use "Compiler Explorer". Choose the language "C", and assembly language "RISC-V rv32gc" to compile a simplified version of C program. We are not going to run it.



<picture of Compiler Explorer page>

Explanation of assembly code line by line:

line 9 (inner product) in the C source is compiled into 47-67 RISC-V 32-bit assembly language.

on the right hand panel (assembly), line 47 loads base-address of A into a0.

line 48,49,50,51 computes i+k*n, stores in a3.

line 53 use "slli" to shift left 2 bits. it align the address with word (32-bit, 4 bytes) boundary.

line 54 add base-address and offset, becomes the effective address of A[i+j*n] in a0.

similary for line 55-61, B[k+j*n] is computed and stored in a1.

line 62,63 multiply A, B together

line 64-66 stores the result to cij.

Now, I want you to study the compiled code but with floating-point as data. You can do that easily in C. Float is 32-bit floating point data in C. Here is the C FP matrix mulitply program. Please note that the program in almost the same as integer version, except the data type becomes float.

```
---------- C program 32-bit floating-point --

#include <stdio.h>

void minit(int n, float m, float* D){
      int i, j;
      for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                  D[i+j*n] = m;
}

void mprint(int n, float* D){
      int i, j;
      for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                  printf("%f ", D[i+j*n]);
      printf("\n");
}

void matmul(int n, float* A, float* B, float* C){
      int i, j, k;
      float cij;

      for( i = 0; i < n; i++)
            for( j = 0; j < n; j++){
                  cij = C[i+j*n];
                  for( k = 0; k < n; k++)
                        cij += A[i+k*n] * B[k+j*n];
                  C[i+j*n] = cij;
            }
}

int main(void){
      float A[9], B[9], C[9];
      int n;

      n = 3;
      minit(n,10,A);
      mprint(n,A);
```

```
      minit(n,20,B);
      mprint(n,B);
      minit(n,0,C);
      mprint(n,C);
      matmul(n,A,B,C);
      mprint(n,C);
}
```

To check that the program is correct, I compiled and run the program. Here is the output:

```
----- float -------------

C:\Users\prabhas\Dropbox\comparch-2023\fmatmul\lcc>fmm
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000
20.000000 20.000000 20.000000 20.000000 20.000000 20.000000 20.000000
20.000000 20.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000
600.000000 600.000000 600.000000 600.000000 600.000000 600.000000
600.000000 600.000000 600.000000
```

Now, your work is the compile floating-point version of matmul() function and study the assembly language output (similar to my explanation). Take screenshot of your "Compiler Explorer" output and submit it. Submit your explanation.

Enjoy !

PS You can try to run it in Venus, you need to include initialise data "minit()" and "main()" too. I did have problem with running it.