CS243 Assignment 7 Solutions

1a. (2pts)

\[ \begin{array}{c}
0 & 1000 \\
\hline
0 & j \\
\hline
k & 1000 \\
\end{array} \]

1b. (3pts)

\[
B[j, k] : \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} j \\ k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

\[
A[j] : \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} j \\ k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

\[
B[j - 1, k - 1] : \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} j \\ k \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix}
\]

1c. (3pts) There are two possible data dependences in this loop.
Reads of \( B[j - 1, k - 1] \) vs. writes of \( B[j, k] \):

\[
\begin{align*}
&j \geq 1 \\
&j \leq 999 \\
&k \geq j \\
&k \leq 999 \\
&j' \geq 1 \\
&j' \leq 999 \\
&k' \geq j \\
&k' \leq 999 \\
&j = j' - 1 \\
&k = k' - 1
\end{align*}
\]
Writes of $B[j, k]$ vs. writes of $B[j, k]$ in distinct iterations of the loop: (these conditions exclude the $i = i'$ condition, see below):

\[
\begin{align*}
  j & \geq 1 \\
  j & \leq 999 \\
  k & \geq j \\
  k & \leq 999 \\
  j' & \geq 1 \\
  j' & \leq 999 \\
  k' & \geq j \\
  k' & \leq 999 \\
  j & = j' \\
  k & = k'
\end{align*}
\]

Representing the $i \neq i'$ condition as an integer linear program requires us to test the above program with each of the following side conditions:

\[
\begin{align*}
  j & \leq j' - 1 \\
  j & \geq j' + 1 \\
  k & \leq k' - 1 \\
  k & \geq k' + 1
\end{align*}
\]

If there is a solution for the program with any of the above side conditions, then there is a data dependence between writes of $B[j, k]$. It was also acceptable just to write $i \neq i'$, or $j \neq j' \lor k \neq k'$, but not $j \neq j' \land k \neq k'$.

1d. (3pts) There is a data dependence for a pair of accesses if there is a solution to the linear program constructed for those accesses in part (c). There is such a solution only for reads of $B[j - 1, k - 1]$ vs. writes of $B[j, k]$. Example iterations where the data dependence occurs:

\[
i: \begin{bmatrix} 10 \\ 10 \end{bmatrix} \\
i': \begin{bmatrix} 11 \\ 11 \end{bmatrix}
\]

1e. (3pts)

\[
\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} j \\ k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]
Any $C$ and $c$ which assigns the same processor to all iterations along a diagonal in the iteration space (i.e. the difference between $j$ and $k$ is constant) is correct.

1f. (3pts)

# Omega Calculator v2.1 (based on Omega Library 2.1, July, 2008):
# S := {[p,j,k]: p = k - j && 1 <= j < 1000 && j <= k < 1000};
#
# codegen S;
for(t1 = 0; t1 <= 998; t1++) {
    for(t2 = 1; t2 <= -t1+999; t2++) {
        s1(t1,t2,t1+t2);
    }
}

2a. (10 pts) The first step to approach this problem is to identify the dependencies for each pair of accesses.

The accesses in this program are:

d. Read access of $A[i,i]$ in statement 1.
e. Read access of $A[j,k]$ in statement 2.
g. Read access of $A[i,k]$ in statement 2.

The loop nest of statement 1 is 2 levels deep. Thus, the iteration space of the accesses of statement 1 is in the two dimensional space over integers, $\mathbb{Z}^2$. Similarly, the loop nest of statement 2 is 3 levels deep. Thus, the iteration space of the accesses of statement 2 is in the three dimensional space over integers, $\mathbb{Z}^3$.

The potential set of dependencies are all pairs of access $a_1$ and $a_2$ such that either one of $a_1$ or $a_2$ is a write access and that there exists a loop index vector $i_1$ in the iteration space of $a_1$, and a loop index vector $i_2$ in the iteration space of $a_2$, such that
the location accessed by \( a_1 \) on iteration \( i_1 \) is the same as the location accessed by \( a_2 \) on iteration \( i_2 \). Additionally, if \( a_1 \) and \( a_2 \) are in the same statement, then \( i_1 \neq i_2 \).

Consider loop 1.

The pair of accesses 1 and 2 are independent if and only if there does not exist \((i,j)\) and \((i',j',k')\) such that:

a. \( j = j' \)

b. \( i = k' \)

c. \( 1 \leq i < 10000 \)

d. \( i + 1 \leq j < 10000 \)

e. \( 1 \leq i' < 10000 \)

f. \( i' + 1 \leq j' < 10000 \)

g. \( i' + 1 \leq k' < 10000 \)

Because these constraints are satisfiable, then the pair of accesses 1 and 2 are dependent. The pair of accesses 1 and 7 are independent if and only if there does not exist \((i,j)\) and \((i',j',k')\) such that:

a. \( i' = j \)

b. \( k' = i \)

c. \( 1 \leq i < 10000 \)

d. \( i + 1 \leq j < 10000 \)

e. \( 1 \leq i' < 10000 \)

f. \( i' + 1 \leq j' < 10000 \)

g. \( i' + 1 \leq k' < 10000 \)

Because these constraints are not satisfiable, then the pair of accesses 1 and 7 are independent. The remaining dependencies can be found similarly by writing out and solving the corresponding integer linear programs.

Now, in order to show that loop 1 is not parallelizable, we determine the affine partition constraints for the given the dependencies. Consider the dependency between accesses 2 and 7. Let \( \langle C_1, c_1 \rangle \) be the affine partition of statement 1 and \( \langle C_2, c_2 \rangle \) the affine partition of statement 2. Then, the affine partition constraint induced by the dependency between accesses 2 and 7 is:

\[
C_2 \ast i_1 + c_2 = C_2 \ast i_2 + c_2
\]
where $i_1 = (i, j, k)$ and $i_2 = (i', j', k')$. The $C_2$ matrix must have 3 columns because the index vector has 3 rows. We initially assume $C_2$ has 1 row regardless of the actual degrees of parallelization. The maximum number of linearly independent solutions to $C_2$ will be the degrees of parallelization. Writing out the constraint in non-matrix form:

$$C_{21} \cdot i + C_{22} \cdot j + C_{23} \cdot k + c_2 = C_{21} \cdot i' + C_{22} \cdot j' + C_{23} \cdot k' + c_2$$

The constraints on the loop index vectors $i_1$ and $i_2$ are $j' = i$ and $k' = k$. Applying this substitution yields:

$$C_{21} \cdot i + C_{22} \cdot j + C_{23} \cdot k + c_2 = C_{21} \cdot i + C_{22} \cdot i + C_{23} \cdot k + c_2$$

Then, writing the constraint yields:

$$(C_{21} - C_{22}) \cdot i + C_{22} \cdot j - C_{21} \cdot i' = 0$$

This requires that:

$C_{21} - C_{22} = 0$
$C_{22} = 0$
$-C_{21} = 0$

The solution that satisfies these constraints is that $C_{21} = C_{22} = 0$ and $C_{23}$ is unconstrained. Consequently, the $C_2$ has 1 linearly independent solution, namely $[0, 0, 1]$. Another solution to $C_2$ is $[0, 0, 5]$ but this is not linearly independent from $[0, 0, 1]$. A single linearly independent solution means that this loop has at most one degree of parallelism. In order to be certain that it has exactly 1 degree, we must consider each of the remaining dependencies. So, next consider the dependency between accesses 2 and 6. Writing out the constraint in non-matrix form:

$$C_{21} \cdot i + C_{22} \cdot j + C_{23} \cdot k + c_2 = C_{21} \cdot i' + C_{22} \cdot j' + C_{23} \cdot k' + c_2$$

The constraints on the loop index vectors $i_1$ and $i_2$ are $j' = j$ and $k' = i$. Applying this substitution yields:

$$C_{21} \cdot i + C_{22} \cdot j + C_{23} \cdot k + c_2 = C_{21} \cdot i + C_{22} \cdot j + C_{23} \cdot i + c_2$$

Then, writing the constraint yields:

$$(C_{21} - C_{23}) \cdot i + C_{23} \cdot k - C_{21} \cdot i' = 0$$

This requires that:

$C_{21} - C_{23} = 0$
$C_{23} = 0$
$-C_{21} = 0$

in addition to the constraints from the dependency between 2 and 7:

$C_{21} - C_{22} = 0$
$C_{22} = 0$
$-C_{21} = 0$
The only solution that satisfies these equations is that $C_{21} = C_{22} = C_{23} = 0$. Consequently, we are now sure that there are exactly zero degrees of parallelization in loop 1 because the only solution to $C_2$ is $[0,0,0]$ which has rank 0. Considering any additional dependencies is not required since we’re at the minimum possible degrees of parallelization.

2b. (5pts)
The pair of accesses 1 and 2 are independent if and only if there does not exist \((j)\) and \((j',k')\) such that:

a. \(j = j'\)
b. \(i = k'\)
c. \(1 \leq i < 10000\)
d. \(i + 1 \leq j < 10000\)
e. \(i + 1 \leq j' < 10000\)
f. \(i + 1 \leq k' < 10000\)

Because these constraints are not satisfiable, then the pair of accesses 1 and 2 are independent.

Note now the loop index \(i\) of the outer loop is being treated as a constant in this set of constraints. The pair of accesses 2 and 6 are also independent by the same reasoning. The pair of accesses 1 and 7 are independent as shown earlier with a stronger set of constraints. The pair of accesses 2 and 7 are independent if and only if there does not exist \((j)\) and \((j',k')\) such that:

a. \(j = j'\)
b. \(k = k'\)
c. \(1 \leq i < 10000\)
d. \(i + 1 \leq j < 10000\)
e. \(i + 1 \leq j' < 10000\)
f. \(i + 1 \leq k' < 10000\)

Because these constraints are not satisfiable, then the pair of accesses 2 and 7 are independent. The pair of accesses 1 and 6 are independent if and only if there does not exist \((j)\) and \((j',k')\) such that:

a. \(j = j'\)
b. \(1 \leq i < 10000\)
c. \(i + 1 \leq j < 10000\)
d. \( i + 1 \leq j' < 10000 \)

e. \( i + 1 \leq k' < 10000 \)

Because these constraints are satisfiable, then the pair of accesses 1 and 6 are dependent. In fact, this pair is the dependent pair of accesses. Now, we determine the affine partition constraints for the given the dependency between 1 and 6. Let \( \langle C_1, c_1 \rangle \) be the affine partition of statement 1 and \( \langle C_2, c_2 \rangle \) the affine partition of statement 2. Then, the affine partition constraint induced by the dependency between accesses 2 and 7 is:

\[
C_2 * i_1 + c_2 = C_2 * i_2 + c_2
\]

where \( i_1 = (j) \) and \( i_2 = (j', k') \). The \( C_1 \) matrix must have 1 column because the index vector \( i_1 \) has 1 row, and the \( C_2 \) matrix must have 2 columns because the index vector \( i_2 \) has 2 rows. The maximum number of linearly independent solutions to \( C_1 \) and \( C_2 \) will be the degrees of parallelization. Writing out the constraint in non-matrix form:

\[
C_{11} * j + c_1 = C_{21} * j' + C_{22} * k' + c_2
\]

The constraints on the loop index vectors \( i_1 \) and \( i_2 \) are \( j' = j \). Applying this substitution yields:

\[
C_{11} * j + c_1 = C_{21} * j + C_{22} * k' + c_2
\]

Then, writing the constraint yields:

\[
(C_{11} - C_{21}) * j - C_{22} * k' + c_1 - c_2 = 0
\]

This requires that:

\[
C_{11} - C_{21} = 0 \\
-C_{22} = 0 \\
c_1 - c_2 = 0
\]

which means that \( C_{11} = C_{21} \), \( c_1 = c_2 \), and \( C_{22} = 0 \). The maximum number of linearly independent solutions of \( C_2 \) is 1 because \( C_{22} \) is 0, so regardless of what \( C_{21} \) is, there will only be one linearly independent solution. Similarly, the maximum number of linearly independent solutions of \( C_1 \) is 1. So, we can pick any solution to the equalities above: \( C_{11} = 1 \), \( C_{21} = 1 \), \( c_1 = 0 \), \( c_2 = 0 \). Consequently, the degrees of parallelization is 1.

Finally, \( p = j \) for both statements 1 and 2.

An SPMD code:

```plaintext
# Omega Calculator v2.1 (based on Omega Library 2.1, July, 2008):
# symbolic i;
#
#
# S1 := {[p,j1,k1]: p = j1 && i+1 <= j1,k1 < 10000};
```

Winter 2014/2015
3. (10 pts) An SPMD code:

```plaintext
# Omega Calculator v2.1 (based on Omega Library 2.1, July, 2008):
# S1 := {[p,i1,j1,k1]: p = i1 + j1 + 2 && 0 <= i1,j1,k1 < 1000};
# #
# S2 := {[p,i2,j2,k2]: p = i2 + j2 && 0 <= i2,j2,k2 < 1000};
# #
# codegen S1,S2;
for(t1 = 0; t1 <= 2000; t1++) {
    for(t2 = max(t1-1001,0); t2 <= min(t1,999); t2++) {
        if (t2 <= t1-2) {
            for(t4 = 0; t4 <= 999; t4++) {
                s1(t1,t2,t1-t2-2,t4);
            }
        }
        if (t2 >= t1-999) {
            for(t4 = 0; t4 <= 999; t4++) {
                s2(t1,t2,t1-t2,t4);
            }
        }
    }
}
```

Note: Any reasonable answer gets full credit.
4a. (3pts) For the inner-most loop, we have 1 miss except for the first iteration for which we have 2 misses. Hence, for the inner-most loop:

$$2 + 9,998 \times 1 = 10,000$$

Since we have 512 iterations for the outer-loop,

$$512 \times 10,000 = 5,120,000$$

We have 5,120,000 misses in total.

4b. (3pts) This is a forall loop with no dependence between iterations, so the inner and outer loops can be exchanged. This causes each iteration of the outer loop to traverse an entire row, rather than an entire column.

```c
int A[10000][512];
for (j = 0; j < 9999; j++) {
    for (i = 0; i < 512; i++) {
    }
}
```

4c. (2pts)

$$5,120,000 / 4 = 1,280,000$$