Massachusetts Institute of Technology

6.035 Computer Language Engineering
Spring 2009

Quiz 3

Thursday, April 30th, 2009

Name: _____

This quiz is open book, open notes. You have 55 minutes to complete it. It contains 17 questions in 9 pages (including this one), totaling 100 points. Before you start, please check your copy to make sure it is complete.
Please write neatly; we cannot give credit for what we cannot read.
Good luck!

1			of 4
2			of 4
3			of 4
4			of 10
5			of 6
6			of 8
7			of 4
8			of 6
9			of 4
10			of 4
11			of 6
12			of 4
13			of 4
14			of 6
15			of 8
16			of 14
17			of 4
			of 100

Consider the following code for Questions 1 and 2.

```
int c = 5;
int d = 0;
int e = 2;
int f = 14;

for i=0 to N {
  int t = 0;

  for j = 0 to N {
    t = t + c;
    x = t + e;
    y = c * e + 25;
    A[x] = y + j;
  }

  c = c + d;
  d = d + 2;
  e = 3 * d + f + 1;
}
```

Question 1. Induction Variables (4 Points)

Identify the basic and dependent induction variables for the inner loop.

Question 2. Induction Variables (4 Points)

Identify the basic and dependent induction variables for the outer loop.

Consider the following loop for Questions 3 and 4:

```
mov $20, %rax
mov A, %r15
mov B, %r16
mov C, %r17

loop:
     mov (%r15, %rax), %r10        # r10 = A[rax]
     imul  %r17, %r10              # r10 = r10*C
     mov (%r16, %rax), %r11        # r11 = B[rax]
     add %r11, %r10                # r10 = r11 + r10
     mov %r10, (%r15, %rax)        # A[rax] = r10
     sub $4, %rax                  # rax = rax – 4
     jz loop
```

Assume our target schedules instructions in-order and does not perform register renaming. It has one memory unit and two ALU units. Assume that a `mov` is either a load or a store (neglect address calculation). All three units are fully-pipelined. Instruction latencies:

Memory:
- load = 2 cycles
- store = 2 cycles

ALU1 and ALU2:
- `add/sub/jz` = 2 cycles
- `imul` = 3 cycles

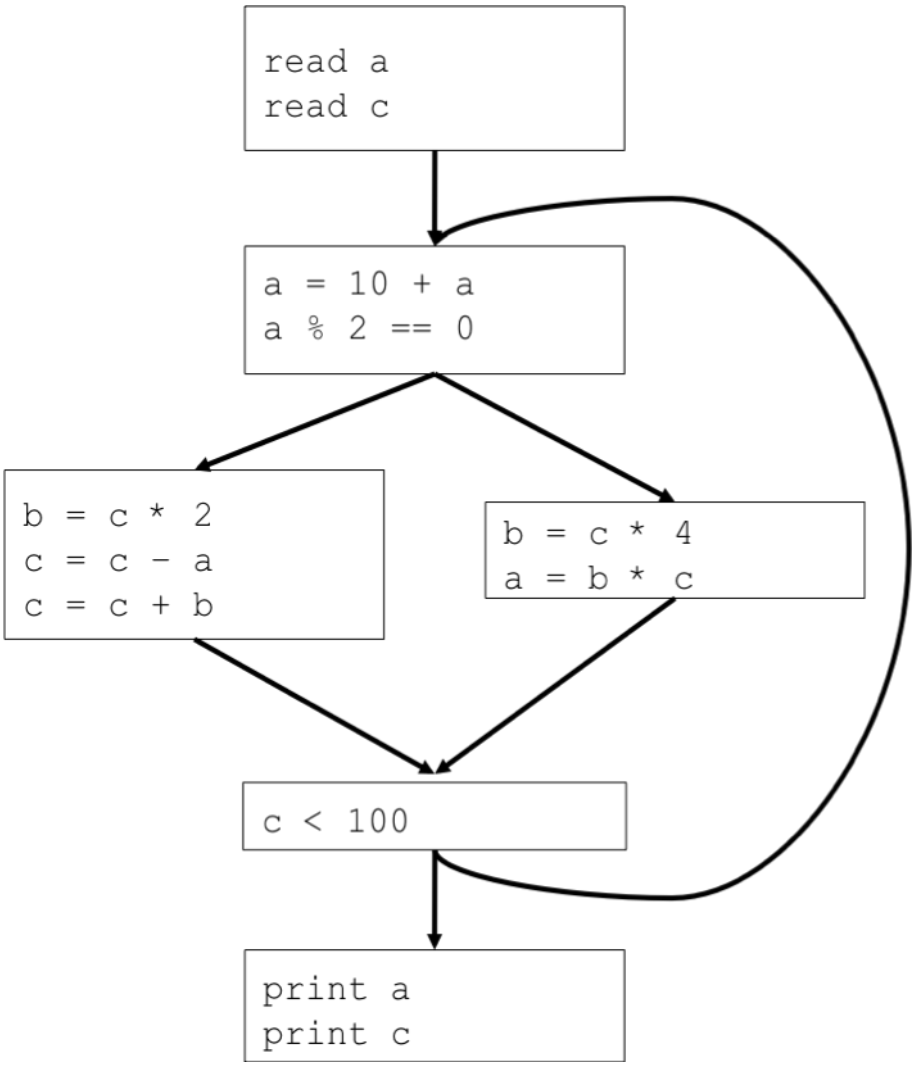
Question 3. Cycles (4 Points)

How many cycles are required for each iteration of the above loop?

Question 4. Unrolling and Scheduling (10 Points)

Create a more efficient version of the loop by unrolling the loop once, removing any unnecessary dependencies, and rescheduling the instructions.

Consider the following CFG for Questions 5-7:



Question 5. Webs (6 Points)

We want to create the maximum number of webs for each variable in the code above. How many webs can we create for each variable? Denote your webs on the code above by adding subscripts to all variable defs and uses. Use the same subscript for defs and uses in the same web.

a has _____ webs.

b has _____ webs.

c has _____ webs.

Question 6. Interference Graph (8 Points)

Draw the interference graph for the webs you defined in Question 5. Label the nodes of your graph with the subscripted variable that the web represents.

Question 7. Colors (4 Points)

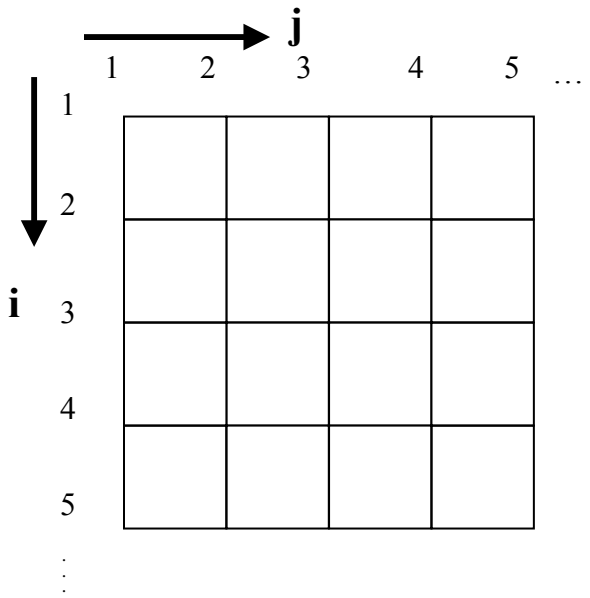
What is the minimum number of colors needed to color your interference graph using the coloring heuristic given in class?

Consider the following loop nest for Questions 8 - 14.

```
for i=1 to n
  for j=1 to n
     A[i, j] = A[i+1, j-1] + A[i-1, j]
```

Question 8. Dependences (6 Points)

Draw the dependences between `A[i, j]` and `A[i+1, j-1]`.



Question 9. Dependence Type (4 Points)

What is the type of dependence between `A[i, j]` and `A[i+1, j-1]`?

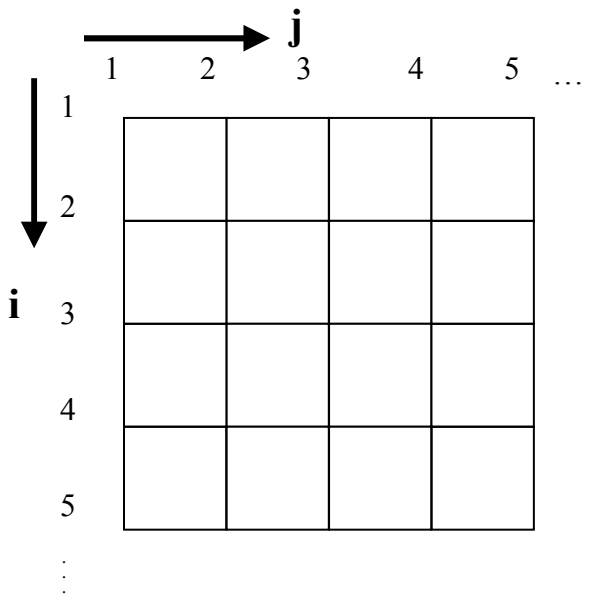
TRUE ANTI OUTPUT

Question 10. Dependence Vector (4 Points)

What is the dependence vector `A[i, j]` and `A[i+1, j-1]`?

Question 11. Dependences (6 Points)

Draw the dependences between `A[i, j]` and `A[i-1, j]`.



Question 12. Dependence Type (4 Points)

What is the type of dependence between `A[i, j]` and `A[i-1, j]`?

TRUE ANTI OUTPUT

Question 13. Dependence Vector (4 Points)

What is the dependence vector `A[i, j]` and `A[i-1, j]`?

Question 14. Parallelization (6 Points)

Select one:

a. Neither loop is parallelizable.

b. The `i` loop is parallelizable.

c. The `j` loop is parallelizable.

d. Both loops are parallelizable.

Consider the following code for Questions 15 - 17.

```
int pow = 1;
int tmp;
int A[N];
…
for I = 0 to N-1 {
    tmp = I * pow;
     A[I] = tmp + A[I+1];
     pow = pow * 2;
}
```

Question 15. Parallelization (8 Points)

Explain all the reasons why this loop is not parallelizable.

Question 16. Parallelization (14 Points)

Rewrite the code so that a parallelizing compiler can efficiently compile the code to a multicore. You can introduce new variables and code, but the values of the variables must be the same after the code executes. Mark the regions of your code that the parallelizing compiler will parallelize.

Question 17. Parallelization (4 Points)

If you execute your modified loop on a single core machine, will it run faster or slower than the original loop? Why?

6.035 Computer Language Engineering
Spring 2010