

Write your answers on another sheet of paper. Homework assignments are to be completed individually. Hand written submissions are fine, but they must be readable. Due at the beginning of class. Total points: 150, 5% of course grade

1. [10 Points] True or False. If a statement is false, explain how so.
 - (a) It is possible to do full-path, context-sensitive analysis for recursive functions.
 - (b) Trace scheduling requires memory dependence profiles.
 - (c) The second Futamura projection compiles a program.
 - (d) The value dependences within a program are a subset of all the memory dependences in a program.
 - (e) Unimodular transformation frameworks subsume the unifying transformation framework built on presburger arithmetic.

2. [20 Points] Interprocedural pointer analysis. Do flow-sensitive, points-to analysis with three different levels of context sensitivity: none, one-level, and full path.

```

int g;
void main () {
    int a,b,c;
    foo(&g, &a); // call site 1 (CS1)
    foo(&a, &c); // CS2
}

void foo(int *p, int *s) {
    bar(p,s); // CS3
}

void bar(int *i, int *j) {
    ...
}

```

Partial answer:

At the beginning of ...	Context Sensitivity		
	NONE	One Level	Path
main	\emptyset	\emptyset	\emptyset
foo	$\{p \rightarrow g, p \rightarrow a, s \rightarrow c, s \rightarrow a\}$??	??
bar	??	??	$\{([CS1, CS3], \{i \rightarrow a, j \rightarrow c, \}), ??\}$

3. [50 Points] Register Allocation. Here is an example program. (Hint: for this problem you will need to rewrite the program five times).

```

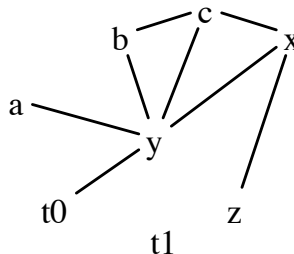
y = ...
t = ...
a = 3 + t
b = a
c = a
x = 40
if (y<x) goto L1
    z = y + c
    x = b + 30
    goto L2
L1:
    z = y + 30
L2:
    t = z + x

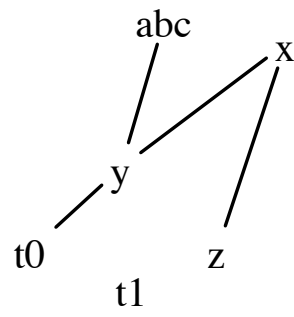
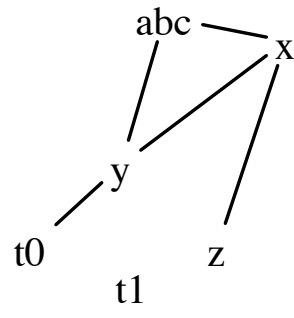
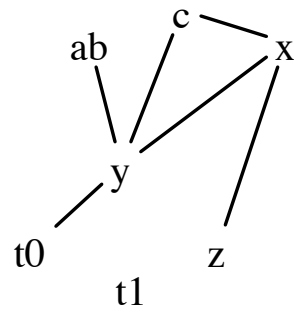
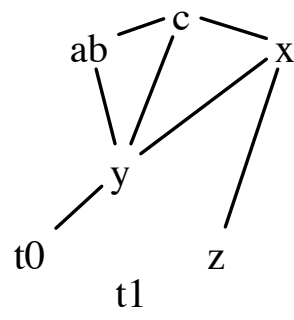
```

- Find and show the webs within the example. Provide a version of the program where each web has been assigned a symbolic register.
- Construct an interference graph and perform as many iterations of coalescing and interference graph construction as is possible. Show all your steps. Coalesce the copy statements in the order they appear in the example.
- Use the Briggs register allocation algorithm to color the graph assuming there are two registers available. Visit the nodes in the graph in alphabetic order if a number of nodes have the same number of interferences edges. Show the code and any new interference graph if spilling occurs.
- What is the final code once the variables have been assigned to registers R1 and R2?

Partial answer:

Here are the interference graphs you will be generating.





4. [30 Points] Instruction Scheduling. The loop shown below

```
for (i=1000; i>0; i--) {
    x[i] = x[i] + s;
}
```

can be converted to MIPS assembly as follows:

```
// body of the loop
Loop:
    L.D      F0, 0(R1)      // F0 = &R1
    ADD.D    F4, F0, F2     // F4 = F0 + F2
    S.D      F4, 0(R1)
    DADDI    R3, R3, #-1    // R3 = R3-1
    DADDI    R1, R1, #-8    // R1 = R1-8
    BNE     R3, R2, Loop
```

Use the following set of interlocks/latencies between different instructions when the second instruction depends on the first instruction. For example, the architecture will interlock for 3 cycles between two DADDI instructions if the instructions are scheduled one after the other. If only one instruction is scheduled in between the two, there will still be two cycles of interlock.

first instruction	second instruction	
	ADD.D	S.D
ADD.D	3	2
L.D	1	0

The body of the loop given above requires 9 cycles to execute. Use list scheduling with the following priorities to find a schedule that only requires 7 cycles:

- Avoid stalls with previously scheduled instructions.
- Does the instruction interlock with any immediate successors?
- How many successors does the instruction have?
- Is the instruction on the critical path?

Show the DAG that you use to perform the scheduling. Assume that the BNE instruction must occur last.

4. [40 Points] Data dependence analysis with Omega.

```
integer i,j,N
real a(0:99)
```

```
for i=0,(N-1) do
  for j=0,(N-1) do
    a(j) = a(i-2)
  endfor
endfor
```

- (a) What is the anti-dependence relation between the read to $a(i-2)$ and the write to $a(j)$?
- (b) Write a Tiny program that results in the following data-dependence relation. There will be other data-dependence relations. The one provided will be one of many.
- $$[i,j] \rightarrow [i+3,j-2] : 0 \leq i \leq N-4 \ \&\& \ 2 \leq j < N$$
- (c) Focusing only on the flow dependence from $c(i)$ to $c(i+1)$, use omega to show that loop fusion is illegal on the following two loops:

```
for i=0,(n-1)
  c(i) = a(i) / 2
endfor
for j=0,(n-1)
  d(j) = 1 / c(j+1)
endfor
```