Test 2

I declare the following statements to be true:

- The work I submit here is entirely my own.
- I have not used any unauthorized aids.
- I have not discussed and will not discuss the contents of this examination with anyone until after the submission deadline.
- I am aware that misconduct related to examinations can result in signi
  cant penalties, including failing the course and suspension.
- Last name: Mai
- First name: Chongju
- Student ID: 150823820
- Signature: Chongju Mai

1

(a)

Stack activation record can save procedure information in a block with certain format, the information include but not limit to return address, local data, arguments ...

(b)

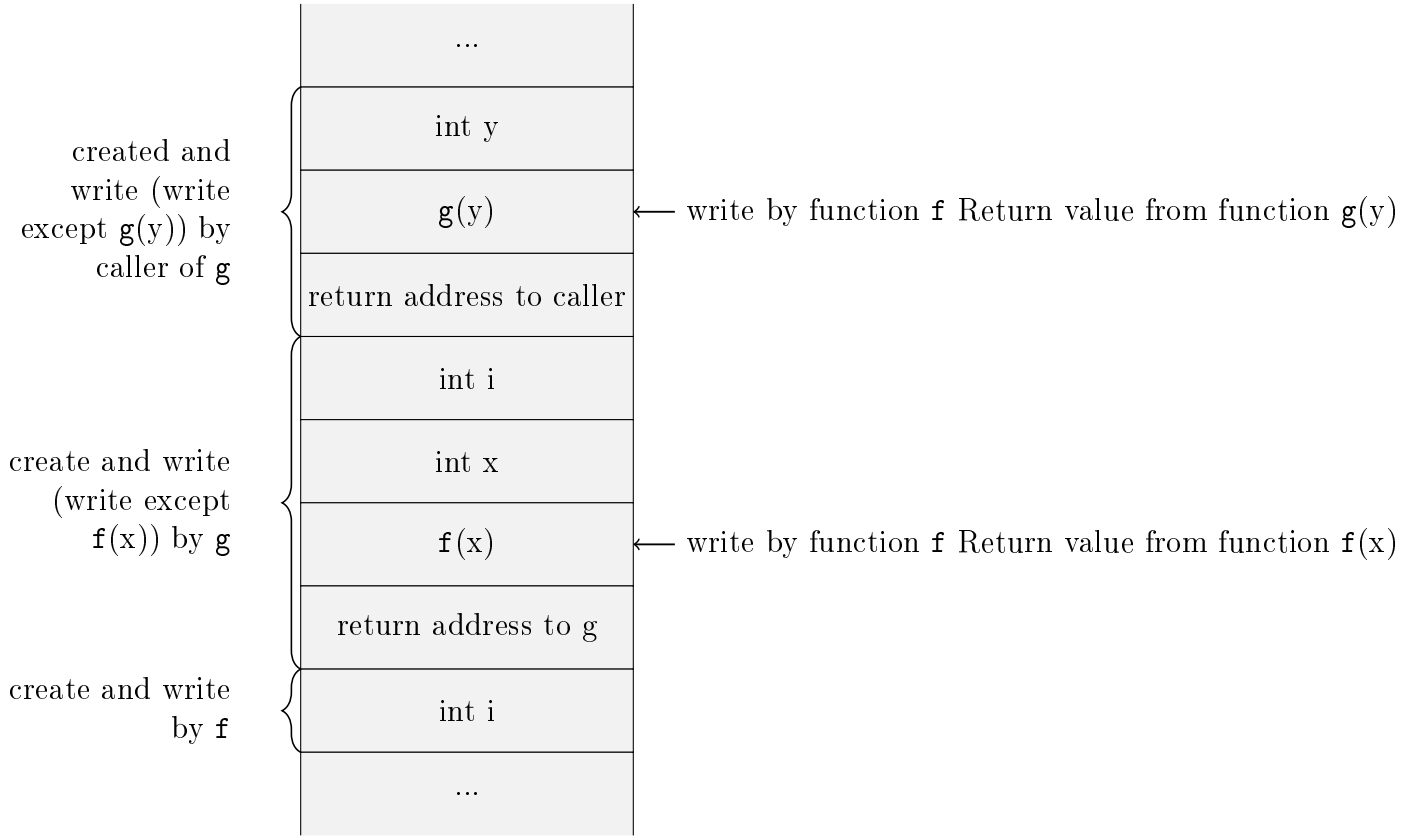
- Local data: save the local data belong to the procedure
- Saved machine status: Save the information about caller procedure. Including return address, content of registers that were used by the calling procedure. This can recover the register data back to original value before return the calling procedure.
- Access link: pointing to another activation record. It is use for nested function.
- Return value: the return value for the current function
- Actual parameter: in case the register does not have enough space to save all the parameters. Extra parameter value will save in this section.

(c)

By using stack, it allow space to be shard by procedure call those duration do not overlap in time. It also allow us to compile code for a procedure in such a way that the relative address of its non-local variable are always the same, regardless of the sequence of procedure calls.

2

The following activation record is growing from top to button.



First four elements (First four from top) belong to function `g` and the other four elements (First four from bottom) belong to function `f`

3

Sequence of three address statements on the left, optimized sequence of three address statement on the right

```
    t1 := a == c
    t2 := a <= b
    ifFalse t1 or t2 goto F
    t3 := b * c
    t4 := a - t3
    a := t4
    t5 := b * c
    t6 := a + t5
    b := t6
    goto E
F:  t7 := 3 * a
    t8 := t7 + 1
    a := t8
    t9 := 3 * a
    t10 := t9 - 2
    b := t10
E:  ...
```

```
    t1 := a == c
    t2 := a <= b
    ifFalse t1 or t2 goto F
    t1 := b * c
    t2 := a - t1
    a := t2
    t2 := a + t1
    b := t2
    goto E
F:  t1 := 3 * a
    t2 := t1 + 1
    a := t2
    t2 := t1 - 2
    b := t2
E:  ...
```

4

(a)

Synthesized attributes are pass form child to its parent. The parent will execute the child node first, once the child node returned, it set its attribute base on the child.
Inherited attributes are pass from its siblings or its parent. The node execute its first child first, once the child node returned, it either pass the attribute from the first child or its own to the second child.

(b)

D → TL {L.in = T.type}
T → `int` {T.type = `'int'`}
T → `real` {T.type = `'real'`}
L → L1,`id` {L1.in = L.in; addtype(`id`.entry, L.in)}
L → `id` {addtype(`id`.entry, L.in)}

(c)

No, because the type is the first to parser. All other id is not visible at that time, id is after type. So we need to pass the type to id. Which it is its sibling. So the type in here must be passing as synthesized attributes.