

Check-In

Write the 3ac for the following code snippet:

```
int v(int a, int b){  
    if (a > 1){  
        if (b < 3){  
            return a + b;  
        }  
    }  
}
```

Check-In Solution

Flipped Wednesday



Administrivia

P5 – new due Monday

Topics

3 Address Code

W7 – Question 1

Convert the following function into 3AC

```
int g;  
int a(int b, int c){  
    if (b){  
        return 0;  
    } else {  
        b = b - 1 * c;  
    }  
    return b;  
}
```

W7 – Question 1 Solution

Convert the following function into 3AC

```
int g;  
int a(int b, int c){  
    if (b){  
        return 0;  
    } else {  
        b = b - 1 * c;  
    }  
    return b;  
}
```

```
fn_a: enter a  
      getarg 1, [b]  
      getarg 2, [c]  
      [tmp1] := [b]  
  
      ifz [tmp1] goto LBL_1  
      setret 0  
      goto end_fn_a  
      goto LBL_2  
LBL_1: nop  
      [tmp1] := 1 MULT64 [c]  
      [tmp2] := [b] SUB64 [tmp1]  
      [b] := [tmp2]  
LBL_2: nop  
      setret [b]  
      goto end_fn_a  
end_fn_a: leave a
```

W7 – Question 2

Convert the following function into 3AC

```
int v(int a){  
    while (a < 2){  
        while (a < 3){  
            a++;  
        }  
        a++;  
    }  
    return a;  
}
```

W7 – Question 2 Solution

Convert the following function into 3AC

```
int v(int a){
    while (a < 2){
        while (a < 3){
            a++;
        }
        a++;
    }
    return a;
}
```

```
fn_v: enter v
        getarg 1, [a]
LBL_1: [tmp1] := [a] LT64 2
        ifz [tmp1] goto LBL_2
LBL_3: [tmp2] := [a] LT64 3
        ifz [tmp2] goto LBL_4
        [a] := [a] ADD64 1
        goto LBL_3
LBL_4: nop
        [a] := [a] ADD64 1
        goto LBL_1
LBL_2: nop
        setret [a]
        goto end_fn_v
end_fn_a: leave v
```

W7 – Question 3

Convert the 3AC procedure into source code

```
fn_k: enter k
      getarg 1, [b]
      [i] = [b]
lbl_1: [t] = [i] LT64 10
      ifz [t] goto lbl_2
      [i] = [i] ADD64 1
      WRITE [i]
      goto lbl_1
      lbl_2: nop
fn_end_k: leave k
```

W7 – Question 3 Solution

Convert the 3AC procedure into source code

```
fn_k: enter k
      getarg 1, [b]
      [i] = [b]
lbl_1: [t] = [i] LT64 10
      ifz [t] goto lbl_2
      [i] = [i] ADD64 1
      WRITE [i]
      goto lbl_1
      lbl_2: nop
fn_end_k: leave k
```

```
void k (int b) {
    int i;
    i = b;
    while (i < 10) {
        i++;
        print i;
    }
}
```

W7 – Question 4

Assume a language that allows for pass-by-reference or pass-by-value parameters. What would the 3AC code look like for a pass-by-reference call? Illustrate with an example.

W7 – Question 4 Solution

Assume a language that allows for pass-by-reference or pass-by-value parameters. What would the 3AC code look like for a pass-by-reference call? Illustrate with an example.

Don't use the brackets around the variable (which indicate a memory lookup) in the generated setarg / getarg

```
void foo(int& arg){
    arg = 3;
}

int main(){
    int p;
    foo(p);
}

fn_foo: enter foo
        getarg 1, arg
        [arg] := 3
end_fn_foo: leave foo

fn_main: enter main
        setarg 1, p
end_fn_main: leave main
```