**Code for Stacks Lecture**

const int globalD=6;

int compute(int x, int y);

int squared(int r);

void main() {

int a, b, c; // These are main’s automatic variables, and will be

a = 10; // stored in main’s frame

b = 16;

c = compute(a,b);

}

int compute(int x, int y) {

int z;

z = squared(x);

z = z + squared(y) + globalD;

return(z);

}

int squared(int r) {

return (r\*r);

}

\_main:

enter #06H

mov.w #000aH,-2[FB] ; a

mov.w #0010H,-4[FB] ; b

mov.w -4[FB],R2 ; b

mov.w -2[FB],R1 ; a

jsr $compute

mov.w R0,-6[FB] ; c

exitd

;## # FUNCTION compute

;## # FRAME AUTO (z) size 2, offset -6

;## # FRAME AUTO (y) size 2, offset -2

;## # FRAME AUTO (x) size 2, offset -4

;## # REGISTER ARG (x) size 2, REGISTER R1

;## # REGISTER ARG (y) size 2, REGISTER R2

.glb $compute

$compute:

enter #06H

mov.w R1,-4[FB] ; x x

mov.w R2,-2[FB] ; y y

mov.w -4[FB],R1 ; x

jsr $squared

mov.w R0,-6[FB] ; z

mov.w -2[FB],R1 ; y

jsr $squared

add.w -6[FB],R0 ; z

add.w \_globalD,R0

mov.w R0,-6[FB] ; z

mov.w -6[FB],R0 ; z

exitd

;## # FUNCTION squared

;## # FRAME AUTO (r) size 2, offset -2

;## # REGISTER ARG (r) size 2, REGISTER R1

;## # ARG Size(0) Auto Size(2) Context Size(5)

.glb $squared

$squared:

enter #02H

mov.w R1,-2[FB] ; r r

mov.w -2[FB],R0 ; r

mul.w -2[FB],R0 ; r

exitd