

Computation Structures

Compilation Worksheet

compile_expr(expr) ⇒ Rx

- Constants: $1234 \Rightarrow Rx$
 - `CMOVE(1234, Rx)`
 - `LD(c1, Rx)`
 - ...
 - `c1: LONG(123456)`
- Variables: $a \Rightarrow Rx$
 - `LD(a, Rx)`
 - ...
 - `a: LONG(0)`
- Variables: $b[expr] \Rightarrow Rx$
 - `compile_expr(expr) ⇒ Rx`
 - `MULC(Rx, bsize, Rx)`
 - `LD(Rx, b, Rx)`
 - ...
 - `// reserve array space`
 - `b: . = . + bsize*blen`
- Operations: $expr_1 + expr_2 \Rightarrow Rx$
 - `compile_expr(expr1) ⇒ Rx`
 - `compile_expr(expr2) ⇒ Ry`
 - `ADD(Rx, Ry, Rx)`
- Procedure call: $f(e_1, e_2, \dots) \Rightarrow Rx$
next lecture!
- Assignment: $a=expr \Rightarrow Rx$
 - `compile_expr(expr) ⇒ Rx`
 - `ST(Rx, a)`

compile_statement(...)

- Unconditional: $expr;$
 - `compile_expr(expr)`
- Compound: $\{ s1; s2; \dots \}$
 - `compile_statement(s1)`
 - `compile_statement(s2)`
 - ...
- Conditional: $if (expr) s1;$
 - `compile_expr(expr) ⇒ Rx`
 - `BF(Rx, Lendif)`
 - `compile_statement(s1)`
 - `Lendif:`
- Conditional: $if (expr) s1; else s2;$
 - `compile_expr(expr) ⇒ Rx`
 - `BF(Rx, Lelse)`
 - `compile_statement(s1)`
 - `BR(Lendif)`
 - `Lelse:`
 - `compile_statement(s1)`
 - `Lendif:`
- Iteration: $while (expr) s1;$
 - `BR(Ltest)`
 - `Lwhile:`
 - `compile_statement(s1)`
 - `Ltest:`
 - `compile_expr(expr) ⇒ Rx`
 - `BT(Rx, Lwhile)`
- Iteration: $for (init; test; incr) s1;$
 - `init;`
 - `while (test) { s1; incr; }`

Problem 1.

Please hand-compile the following snippets of C code into equivalent Beta assembly language statements. Assume that memory locations have been allocated for the all C variables with labels that corresponds to the variable names. So to load the value of the C variable `a` into register `R3`, the appropriate assembly language statement would be `LD(R31, a, R3)`. And to store the value in `R17` to the C variable `b`, the appropriate assembly language statement would be `ST(R17, b, R31)`. Similarly, assume that memory locations have been allocated for each C array, with a label defined whose value is the address of the 0th element of the array.

(A) `a = 42;`

(F) `x = y[3] + y[12];`

(B) `c = 5*x - 13;`

(G) `if (b == 0 || b < min) {
 min = b;
} else {
 too_big += 1;
}`

(C) `y = (x - 3)*(y + 123456);`

(D) `if (a == 3) b = b + 1;`

(H) `sum = 0;
i = 0;
while (i < 10) {
 sum = sum + i
 i = i + 1;
}`

(E) `a[i] = a[i-1];`

Problem 3.

Which of the following Beta instruction sequences might have resulted from compiling the following C statement? For each sequence describe the value that does end up as the value of y.

```
int x[20], y;  
y = x[1] + 4;
```

- A. LD (R31, x + 1, R0)
ADDC (R0, 4, R0)
ST (R0, y, R31)

- B. CMOVE (4, R0)
ADDC (R0, x + 4, R0)
ST (R0, y, R31)

- C. LD (R31, x + 4, R0)
ST (R0, y + 4, R31)

- D. CMOVE (4, R0)
LD (R0, x, R1)
ST (R1, y, R0)

- E. LD (R31, x + 4, R0)
ADDC (R0, 4, R0)
ST (R0, y, R31)

- F. ADDC (R31, x + 1, R0)
ADDC (R0, 4, R0)
ST (R0, y, R31)