R1 starts as pointer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Line | Binary Instruction | Human Readable |  |  |
| 0: | bf 16 00 00 00 00 00 00 | r6 = r1 |  |  |
| 1: | b7 01 00 00 00 00 00 00 | r1 = 0 |  |  |
| 2: | 61 62 24 00 00 00 00 00 | r2 = \*(u32 \*)(r6 + 36) |  |  |
| 3: | 56 02 04 00 06 00 00 00 | if w2 != 6 goto +4 <LBB2\_2> |  |  |
| 4: | 85 00 00 00 07 00 00 00 | call 7 |  |  |
| 5: | bc 01 00 00 00 00 00 00 | w1 = w0 |  |  |
| 6: | 67 01 00 00 20 00 00 00 | r1 <<= 32 |  |  |
| 7: | 77 01 00 00 20 00 00 00 | r1 >>= 32 |  |  |

**Using Static Single Assignment form**

Register names r0-r10 will now be differentiated based on repetition of assignment, ie r11 = first use of r1, r12 = second use… I could also name them based on which line they show up in, but that might be more helpful in a larger program if I need to reference where a specific type equation came from.

* Pre Given Info: [r11] = &int
* Line 0: [r61]=[r11]
* Line 1: [r12]=int
* Replacing pointer arithmetic in line 2 with variable
  + Since the opcode is 61, it is defined as a memory instruction, can we assume that the src+offset portion maintains type of src?
  + [y1]=[r61+36] = [r61] ?
* Line 2: [r21]=[\*(y1)]
* Line 3: [w21] = [r21]
  + But since we don’t “know” that r2 is an int, is this typing of subregister and register valid? Even though in the next line it’s going to show that w2 subregister will be forced as type int, but with the SSA assignments, w21 and w22 arent the same register anymore
* Line 3: [6] = int
* Line 3: [w21] or [w22]? = [6]
  + Since this is the first assignment of w2 inside the program proper, but the second time it shows up on the type variable equation list I’ve made, is this still w21? If it was w22, it would be immediately typable, so I guess that’s nice?
* Line 4: [r01] = return type of helper function 7 = int
  + Based on the list of helper functions, I’m going to assume fun7 is just literally the 8th (7 with 0 indexing?) function on the list which has the following signature:
    - *u32 bpf\_get\_prandom\_u32(void)*
    - So it’s looking like a return type of int in our limited typing scheme
* Line 5: [w01] = [r01]
  + Same concern as Line 3 register/subregister typing
* Line 5: [w11] =[w01]
* Line 6: [r13] = [w11]
* Line 6: [r14] = [r13]
* Line 7: [r15] = [r14]