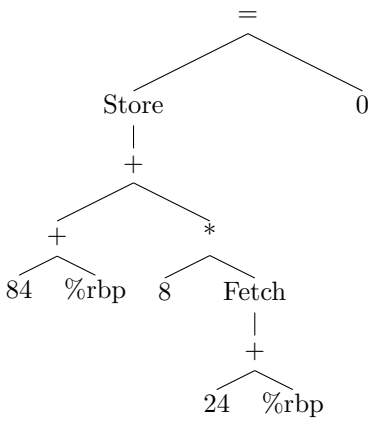
UW Computer Science and Engineering
CSE401: Introduction to Compilers
Robert R. Henry
Dynamic Programming with Tree Pattern Matching Code Generators
Bottom Up Rewrite Systems (BURS)
05Mar2014

The C source code:

```
int A[...];     // A presumed to start 84 bytes from %rbp
int i = ...;    // i presumed to start 24 bytes from the %rbp
A[i] = 0;
```

The subject computation tree we want to generate code for:

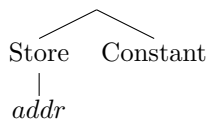
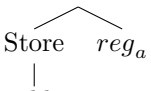
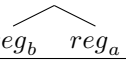
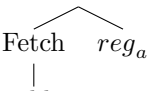
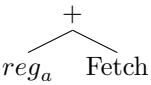
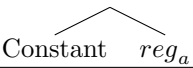
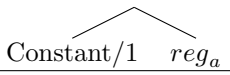
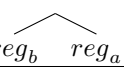
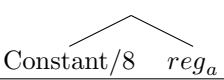
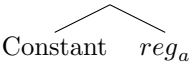


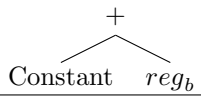
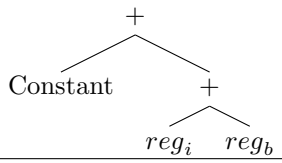
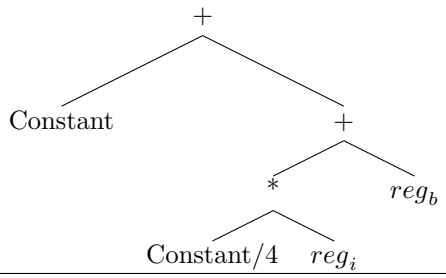
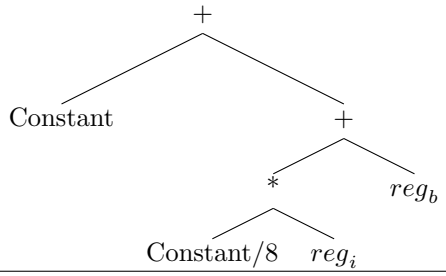
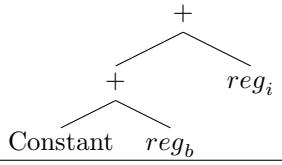
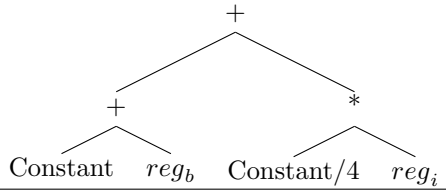
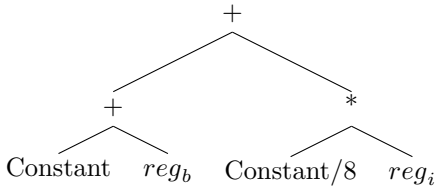
We assume that constants always appear on the left hand side of the operator.

Here, we simply count instructions as our metric, and minimize the number of instructions generated.

We could also use other plausible metrics, such as istream encoding bytes, machine cycles, and power. However, counting cycles and power is likely to be difficult to do, since they vary depending on the context of the instruction, such as the instruction's location in the cache, or super-secret undocumented manufacture private information. Even counting the encoding bytes is difficult, since on x86_64 using some registers costs more to encode than using other registers, and we like to defer register allocation until a follow on pass of the compiler.

The instruction tree patterns (the distinguished symbol is *Root*):

Number	Replacement	Pattern	Cost	Assembly
1	Root		1	movq $Constant, *addr*
2	Root		1	movq reg_a, *addr*
3	reg_a	reg_b	1	movq reg_b,reg_a
4	reg_a	Constant	1	movq $Constant, reg_a
5	reg_a		1	movq *addr*,reg_a
6	reg_a	*addr*	1	leaq *addr*,reg_a
7	reg_a		1	addq reg_b,reg_a
8	reg_a		1	addq *addr*,reg_a
9	reg_a		1	addq *addr*,reg_a
10	reg_a		1	addq $Constant,$reg_a$
11	reg_a		1	incq reg_a
12	reg_a	Constant/0	1	xorq reg_a, reg_a
13	reg_a		1	mulq reg_b,reg_a
14	reg_a		1	aslq $3,$reg_a$
15	reg_a		1	mulq $Constant,$reg_a$

Number	Replacement	Pattern	Cost	Assembly
16	*addr*	reg_b	0	(reg_b)
17	*addr*		0	Constant(reg_b)
18	*addr*		0	Constant(reg_b,reg_i,1)
19	*addr*		0	Constant(reg_b,reg_i,4)
20	*addr*		0	Constant(reg_b,reg_i,8)
21	*addr*		0	Constant(reg_b,reg_i,1)
22	*addr*		0	Constant(reg_b,reg_i,4)
23	*addr*		0	Constant(reg_b,reg_i,8)