Notes and Solutions
6.001 Spring 2007 - recitation 22

register machines, not procedure call

register machines don't support ANY abstractions. ie all abstractions must be maintained by the programmer. do help with this, some conventions:

inputs generally in argN registers, outputs generally in result register
when a block of code is done (successfully computed its output), it does:
(goto (reg continue))
the continue register contains where to go next (often a (halt) instruction).

```
double        ; starting label
 (assign result (op *) (reg arg0) (const 2)) ; only 1 op, tags
 (goto (reg continue))  ; we're done, jump to where we're supposed to go next
```

problem 1

```
func
 (assign result (op *) (reg arg0) (reg arg0))
 (assign result (op +) (reg result) (reg arg1))
 (goto (reg continue))

;could use temporary register(s) (named anything). registers are a
; commodity in short supply; try to get by with less
```

problem 2

```
abs
 (assign result (reg arg0))
 (test (op >) (reg arg0) (const 0))
 (branch (label positive))
 (assign result (op *) (reg result) (const -1))
positive
 (goto (reg continue))
```

or

```
abs
 (test (op <) (reg arg0) (const 0))
 (branch (label negative))
 (assign result (reg arg0))
 (goto (reg continue))
negative
 (assign result (op *) (reg arg0) (const -1))
 (goto (reg continue))
```

```
; either way works.. one is more code-efficient

problem 3

infinite-loop
 (goto (label infinite-loop))

; shortest method.. many others

problem 4

foo
 (test (op <) (reg arg0) (reg arg1))
 (branch (label foo-done))
 (assign arg0 (op -) (reg arg0) (reg arg1))
 (goto (label foo))
foo-done
 (assign result (op =) (reg arg0) (const 0))
 (goto (reg continue))

divisible?

(define (divisible? x y)
  (if (< x y)
      (= x 0)
      (divisible? (- x y) y)))

problem 5

sum-digits
 (assign result (const 0))
sum-digits-top
 (test (op <) (reg arg0) (const 10))
 (branch (label last-sum))
 (assign tmp (op remainder) (reg arg0) (const 10))
 (assign result (op +) (reg tmp) (reg result))
 (assign arg0 (op quotient) (reg arg0) (const 10))
 (goto (label sum-digits-top))
last-sum
 (assign result (op +) (reg result) (reg arg0))
 (goto (reg continue))

problem 6

reduce-to-digit
 (assign num (reg arg0))
 (assign old-continue (reg continue))
 (assign continue (label when-done))
reduce-top
```

```
 (goto (label sum-digits))
when-done
 (test (op <) (reg result) (const 10))
 (branch (label reduce-done))
 (assign arg0 (reg result))
 (goto (label reduce-top))
reduce-done
 (assign result (op cons) (reg num) (reg result))
 (goto (reg old-continue))
```

problem 7

```
reduce-to-digit
 (save arg0)
 (save continue)
 (assign continue (label when-done))
reduce-top
 (goto (label sum-digits))
when-done
 (test (op <) (reg result) (const 10))
 (branch (label reduce-done))
 (assign arg0 (reg result))
 (goto (label reduce-top))
reduce-done
 (restore continue)
 (restore arg0)
 (assign result (op cons) (reg arg0) (reg result))
 (goto (reg continue))
```