

# Linguagens de Programação

---

Fabio Mascarenhas - 2013.1

<http://www.dcc.ufrj.br/~fabiom/lp>

# Threads

---

- Podemos representar uma thread com uma tripla de um valor pendente que vai ser passado pra thread, seu stack pointer, e uma continuação
- Agora podemos manter nas computações um *relógio*, que conta quantos tiques temos até trocar de thread, e uma fila de threads em espera
- Onde ficaria o escalonador? Uma ideia é fazer `bind` agir como um escalonador, decrementando o relógio e trocando de thread quando necessário

```
def bind(a1: Acao, f: Valor => Acao): Acao = k =>
  a1(v => (tick, sp, ths, mem) =>
    if (tick > 0) f(v)(k)(tick - 1, sp, ths, mem)
    else {
      val (tv, tsp, tk, nth) = changeThread(v, sp, f, k, ths)
      tk(tv)(TICKS, tsp, nth, mem)
    })
```

# Threads, take 2

---

- Uma implementação de threads através de uma mudança em `bind` poderia ser feita sem expor as continuções para o interpretador
- Usando continuções podemos deixar `bind` como estava, e embutir o escalonador nas *primitivas*, pois elas controlam “o que fazer depois” pela escolha de usar *k* ou não
- É como o que fizemos com as exceções
- Se a primitiva faz o relógio chegar a 0 então guardamos a continuação dela e usamos a da primeira thread em espera