Successive recastings

```
addition :: Int -> Int -> Int
addition x y = (x + y)
```

Haskell

```
addition_r8m :: GHC.Types.Int -> GHC.Types.Int -> GHC.Types.Int
[GblId, Arity=2, Str=DmdType]
  \ (x_a9l :: GHC.Types.Int) (y_a9m :: GHC.Types.Int) ->
    src<stages.hs:3:1-22>
    GHC.Num.+
      @ GHC.Types.Int
      GHC.Num.$fNumInt
      (src<stages.hs:3:17> x_a9l)
      (src<stages.hs:3:21> y_a9m)
```

Core

Stg

```
addition_r8m :: GHC.Types.Int -> GHC.Types.Int -> GHC.Types.Int
[GblId, Arity=2, Str=DmdType, Unf=OtherCon []] =
    sat-only \r srt:SRT:[(r9o, GHC.Num.$fNumInt)] [x_smq y_smr]
        src<stages.hs:3:1-22>
        src<stages.hs:3:17>
        src<stages.hs:3:21> GHC.Num.+ GHC.Num.$fNumInt x_smq y_smr;
```

```
...
  //tick src<stages.hs:3:1-22>
  //tick src<stages.hs:3:17>
  //tick src<stages.hs:3:21>
...
CmG:
  R2 = GHC.Num.$fNumInt_closure;   // CmmAssign
  I64[(old + 32)] = stg_ap_pp_info;   // CmmStore
  P64[(old + 24)] = _smo::P64;   // CmmStore
  P64[(old + 16)] = _smp::P64;   // CmmStore
  call GHC.Num.+_info(R2) args: 32, res: 0, upd: 8;   // CmmCall
...
```

Cmm

```
...
_cmG:
   movq %r14,%rax
   movl $GHC.Num.$fNumInt_closure,%r14d
   movq $stg_ap_pp_info,-24(%rbp)
   movq %rax,-16(%rbp)
   movq %rsi,-8(%rbp)
   addq $-24,%rbp
   jmp GHC.Num.+_info
...
```

x64 assembly