Published: 5/5/67

Identification

Conditions that necessitate dope and prologue.

James F. Gimpel

Purpose

Not all dope and prologue is necessary. This section gives the conditions under which it is needed.

Discussion

Dope is information which describes data, as distinct from the data itself. Both differ from a specifier which describes what data goes with what dope. It is dope which permits one to say, for a parameter alpha,

```
dcl    alpha bit (*);
```

but dope is not always needed. In fact, most dope can and will disappear.

Dope is created for two purposes

1. To obtain information about the aggregate at runtime that the compiler could not know at compile time.
2. To pass to a called procedure.

But if an aggregate is not adjustable, purpose 1 does not apply. If, moreover, neither the aggregate nor any of its subaggregates is used as an argument, then purpose 2 does not apply. Note that, if a string subelement of a structure is passed as an argument, dope is created at the call

(it is as easy to do this as to access the original dope) so that an aggregate's dope is not needed simply because a single string subelement was passed as an argument.

Example:

```
declare  1 alpha,
                2 beta, 3 betal fixed, 3 beta2 fixed,
                2 gamma bit (10),
         p ptr;
```

1. Does not force dope

```
p = addr (alpha);
call delta (p);
```

Forces dope for alpha

```
call delta (alpha)
```

2. Does not force dope

```
p = addr (alpha.gamma);
call delta (p)
```

Forces dope for gamma (on the fly) but not for alpha

```
call delta (alpha.gamma);
```

3. Does not force dope

```
p = addr (alpha.beta);
call delta (p);
```

Forces dope for alpha

```
call delta (alpha.beta);
```

4. Does not force dope

```
call delta (alpha.beta.betal);
```

In the example given, by passing pointers rather than aggregates we save 12 locations of dope (it is not

unusual to have more dope than data), at most six executable instructions to set up the specifier, and four stack locations. These are modest gains but the example cited is only a small structure. The dope vector for the file system aggregate known as the core map is, at this writing, 131 locations long. Any procedure which (currently) so much as refers to the core map gets created for it, this huge dope vector whether it needs it or not. In the near future it will go away unless the core map itself or one of its subaggregates is used as an argument.

Another gain in passing pointers rather than aggregates is the ease of accessing by the called procedure but this is more properly the domain of Section BN.9.01.

Directly-Addressable, Adjustable Data

It occurs not infrequently that an entire aggregate is directly addressable* but adjustable. For example

```
declare  alpha (0:n) fixed;
```

is such an aggregate. Dope is not needed to access alpha if alpha is not a parameter. If, in addition, alpha is not used as an argument to some other procedure, there would never seem to be any need to have dope for alpha. But the compiler produces code to access dope when an hbounds call is given. These and other difficulties will probably necessitate the keeping of dope for all adjustable aggregates but this is uncertain at the moment.

---

* See Section BN.9.01a for a definition of directly-addressable.