Published: 11/16/66

Identification

Special Built-In Functions
D. B. Wagner, R. M. Graham, and J. D. Harkins

Purpose

A number of special built-in functions need to be added to PL/I if it is to be used for general system programming. These functions are described in this section.

Processor Interlocking

Two instructions in the GE645 involve a read-alter-rewrite cycle and so may be used in the programming of processor interlocks. These are: aos (add one to storage), and stac (Store A conditionally). The following built-in functions provide a way to use these instructions in PL/I-compiled programs.

The references are in the form:

```
result = aos(y)
result = stac(x,y)
```

Here x is a 36-bit string, y is an aligned 36-bit string, and result is a 4-bit string giving the condition of the machine zero, negative, carry, and overflow indicators resulting from the operation performed.

The code generated for stac is roughly as follows:

```
lda     x
stac    y
...     (put indicators into result)
```

and for aos is simply

```
aos     y
...     (put indicators into result)
```

Allocation of Varying Strings

The Multics system programmer needs to be able to control the allocation of varying strings just as he does other types of data. The external procedure allocate_varying is used to specify dynamically the area into which a varying string is placed. It is called by:

```
call allocate_varying (name,area)
```

This would presumably be implemented by copying the current value of the string into the new area and modifying the specifier.

Size Functions

Storage allocation in the file system depends upon the ability to recover gracefully from an error of the form "not enough room in the area". Since some sensitive parts of the file system cannot take SIGNAL's (the standard action for this error is SIGNAL AREA), special built-in functions are needed in order to detect before an allocation whether it will be successful.

1. The function (which need not be built-in) max_stratum takes an area as argument and returns a fixed-point binary number of precision 18 giving the size in machine words of the largest item which can be allocated into the area.

2. The built-in generic function size takes any variable-name (but this will normally be the name of a based variable) and returns a fixed-point binary number of precision 18 giving the size in machine words of the data (not dope or specifier) associated with the variable. The function value is undefined for an argument whose storage is not contiguous, such as a cross-section or defined variable.

Real Time Clock

A real time clock is attached to the GE645. Executing the machine instruction RCCL, with the appropriate effective address, will load the A-Q registers with a 72 bit integer which is the number of microseconds since 0000 GMT, January 1, 1901. The built-in abnormal function "clock_", with no arguments, should return a 71 bit fixed binary number as its value. This function should compile the single instruction,

```
RCCL <clock_>|0
```

The system will work some magic so that the link established for <clock_>|0 will end up with the appropriate effective address for the RCCL instruction. For example, the statements,

```
dcl x fixed binary (71);
x = clock_
```

should compile as

```
RCCL <clock_>|0
STAQ x
```