Identification

System Controller Addressing
J. H. Saltzer

Discussion

In a 645 processor operating in appending mode all instructions
which generate operand addresses have these addresses
run through the appending (page and segment) hardware,
and through the interlace and port selection hardware.
This statement applies even to the instructions which
address the system controllers themselves, rather than
the memory addresses contained within the system controllers.
Several such instructions exist, to manipulate special
registers located within a system controller -- the interrupt
cells, the interrupt mask register, the calendar clock
register, and the alarm clock register. By hardware interface
convention, the absolute address resulting after appending,
interlace, and port selection need merely lie anywhere
within the system controller containing the register of
interest for the instruction to work properly.

For a program which wishes to, for example, read the contents
of the calendar clock register in system controller 3,
it is a non-trivial task to generate a segment-number
word-number pair which results in an absolute address
lying within system controller 3. Even if the program
somehow discovers a location within some segment which
works, it may stop working as soon as the file system
reloads the page containing the magic location into another
absolute address.

To provide a way of addressing system controllers, the
following strategy is followed in Multics:

    a.   For each system controller, dedicate one
           64-word page whose base address lies in that
           system controller. (It is not obvious that
           this is possible with interlace, since one
           might guess that interlace would cause all pages
           to be based in the same system controller. The
           600-line interlace scheme is unusual in that it
           does not cause all addresses which are k(mod)8
           to lie in system controller k. Thus strategy
           a. is possible.)

b.   Make up a page table for which the first m
      entries point to the m pages dedicated above.
      The remaining entries in the page table are
      filled with directed faults.

c.   Include a segment descriptor word pointing to
      this page table in the template descriptor segment.

Now, if the segment descriptor word is in position scas_segno,
one may generate an address lying in the appropriate system
controller by, for example,

```
eabbb     scas_segno
smcm      bb|k*64
```

which sets the interrupt mask in controller k.

It should be pointed out here that the programmer does
not generally know the value of k to use to get to the
register he wants -- the appropriate value changes every
time the hardware configuration is modified. ITS pointers
of the form scas_segno|k*64 are therefore placed in the
system communication segment at initialization or reconfiguration
time, by a program which knows the current hardware configuration.
The programmer now writes the instruction

```
smcm      <scs>|[mask_ptr],*
```

using indirect addressing through the symbolic name of
the appropriate ITS pointer to get at the register he
wants. A complete description of the system communication
segment, giving the list of symbolic entry names, and
the system controller addressing segment are found in
BK.4.01-3.

An appropriate hardware modification to the 645 processor
to simplify system controller addressing would be to discover
during operation decoding that the instruction is a system
controller addressing instruction, and instead of processing
the effective address of the instruction normally, consider
the low-order three bits of the word number of the effective
address to be port selection bits. Thus the first instruction
above would be replaced simply by

```
smcm      k
```

and the second, using indirect addressing would still be

```
smcm      <scs>|[x],*
```

but location x of scs now contains just the number k in the
address field.