Is 32 Bits of Address Too Much?

James C. Brakefield
Technology Incorporated
San Antonio, Texas

Several forthcoming microprocessors (68000, 16000) offer a 24 bit address (16 million bytes of memory space) with eventual expansion to a 32 bit address (4 billion bytes). No one doubts that it is possible to build and support that much virtual memory capacity. Several mini and maxi-computers do.

The question is whether it is more useful to allocate some of the 32 address bits to some other function.

One such allocation is to encode the properties of the operand that an address references. The two major properties are size and format.

The basic formats:

Unsigned integer
Signed integer
Floating point
Others (possibly via an exception mechanism)

The possible sizes:

1 to 64 bits

A simple allocation is: 2 bits for format, 6 bits for size, 3 bits for byte offset; leaving 21 bits for the address (2 megabyte address space). It is felt that this is too small an address space. Thus a more compact allocation is desirable.

One compacting scheme is to limit the sizes to powers of 2 with the data aligned on the power-of-2 boundary. This is encoded by a first-one-bit scheme. The address field is scanned low order to high order looking for a one bit. The location of the first one bit gives the power of 2. The rest of the address field gives the necessary and sufficient address bits.

XXXXXXXXXX1	1 bit object with bit address
XXXXXXXXX10	2 bit object with address
XXXXXXX1000	byte object with byte address
XXXX1000000	64 bit object with longword address

The encoding takes one bit in addition to the address field. With 2 bits for format, 1 bit for size, and 3 bits for byte offset leaves 26 bits for the address or 64 Megabytes address space.

Since a 1, 2, or 4 bit floating point or a 1 bit signed number are not very useful, a similar first-one-bit scheme can be used to encode the format saving a bit:

XXXXXXXX1	unsigned	1, 2, 4, 8, ... sizes
XXXXXXX10	signed	2, 4, 8, ... sizes
XXXXXX100	floating point	4, 8, 16, ... sizes
XXXXX1000	other	

The author prefers a little more control over size, so a 3X bit is included yielding the following:

(1) 3X bit
(1) leading one bit encoding of format
(1) leading one bit encoding of size
(29) address bits for a 64 Megabyte address space

Unsigned sizes	1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, ...
Signed sizes	2, 4, 6, 8, 12, 16, 24, 32, 48, 64, ...
Floating sizes	4, 8, 12, 16, 24, 32, 48, 64, ...

The major advantage of making the address into a more complex object which encodes properties of the operand is that subroutines can be called with a variety of operand types. This should either reduce the number of subroutines, reduce their complexity, or increase their generality.

The first-one-bit encoding has an advantage that sizes can be larger than 64 bits, even to the point that the entire memory space can be specified. This is a possible way to handle records and strings as will as for memory management. This feature would probably also be implemented via an exception mechanism.

References:

1) Memory-management units help 16-bit uP's to handle large memory systems, Jackson Hu et al., p. 128-135, Electronic Design 9, April 26, 1980 (Z8000).
2) A Microprocessor Architecture for a Changing World: The Motorola 68000, Ed Stritter & Tom Gunter, p. 43-52, IEEE Computer Magazine, Feb. 1979
3) Architecture of 16-bit micro underpins VLSI computing, Dan O'Dowd et al., p. 171-176, Electronic Design, June 7, 1980 (NS16000)
4) 16-bit microprocessor enters virtual momory domain, Yohav Lavi et. al., p. 123-129, Electronics, April 24, 1980 (NS16000)
5) The INTEL 8087 numeric dataprocessor, John Palmer, p. 887-893, 1980 National Computer Conference, Vol 49