ANY-1 Design Rationale

© 2021 Robert Finch

## Instruction Size

There is some consternation over the size of instructions. It is highly desirable to have a fixed size instruction as that eases the implementation considerably for fetching and executing more than one instruction per clock. the Given the desired number of registers (32 or more) and the vector nature of the instruction set squeezing everything into a 32-bit instruction just does not work for many instructions. The next simplest to implement instruction size is 64-bits. Using a full 64-bits to represent instructions is too many for most instructions. So, what is proposed for ANY-1 is to use 64-bit parcels of which only the lower 48-bits are used. This would allow the instruction cache to implement only 48 of 64-bits.

The processor addresses 64-bit instruction parcels and main memory holds a full 64-bit parcel.

### Issues

If only 48 of 64-bits are implemented then it is not possible to store 64-bit constants in the instruction memory. Constants larger than 48-bits may be stored in the read-only data section.

## Number of Registers

Implicit in the vector machine is the use of a unified register file. Vector registers may contain integer or floating-point data. Similarly, the general register file may contain integer or floating-point data. A 32-register machine has been found to be a good match for modern compilers. That is 32 registers of each class. Since there are two basic classes of register (integer and floating-point) a 64-register machine would be a good compiler fit. Often in designs registers specs are limited to five bits and the instructions determine the register file (integer or floating-point) in use. Things are done this way only to conserve bit usage in the instruction. With a wider instruction format it is not necessary to do this and six bit register specs can be used.

## Instruction Set

### Jumps

Jumps and calls can be handled with a single instruction due to the wide instruction format. The constant value allows a range of +/- 4GB once shifted. The return address (the address of the next instruction) is stored in the target register. A jump is then made to the sum of a register and twenty-eight bit constant shifted left four times (\*16). Jumps and calls will typically be to the start of a cache line which is 32-byte aligned. Meaning the five LSBs of the target address are zero.

### Branches

Because of the wider instruction format, there is room in the instruction to include an additional register spec for branches which is good because it allows a branch to register. Branching to a register allows conditional return from subroutine and computed branch targets. The branch target is a branch to register plus displacement shifted left three times (\*8). If the target register is specified as x63 then the current value of the program counter is used. This allows program counter relative branching.