Generic ASTOC11
James C. Brakefield

ASTOC11 stands for **A**ccumulator/**St**ack **o**riented **c**omputer with PDP-**11** addressing modes.
An ASTOC11 ISA has the following instruction fields:
Instruction length can be implied or encoded by a length field. Typically one byte (dual stack op-code), two byte (op-code and stack reference) and three bytes for three operand instructions (two stack references). The top of data stack (TOS) is usually one of the operands.

Typically there is a five or six bit op-code. Additional op-code bits on three byte instructions.
A replace bit to indicate result goes to the stack reference rather than to the TOS.
A push/pop bit which can push the result onto the data stack or pop the data stack if the result goes to the stack reference.
A return bit to enable a subroutine return in addition to the other work of the instruction.
A data size field is needed. Four sizes are preferred and two sizes on reduced implementations.
A stack reference mode field uses a subset of the PDP-11 addressing modes[1]. Four modes is the minimum and eight or more is possible.
A stack reference field which contains a two bit stack indicator and a two or more bit offset from the associated stack pointer. The stacks are: the return stack, the data stack, a frame pointer and a global/thread/base area. The frame area should support 16 frame entries, the data stack eight entries, the return stack four and the global area also four.
Some combinations of data size and addressing mode may indicate that the stack reference field is an immediate value operand. In that case most of the stack reference field will encode small constants and the remaining codes used to indicate the number for following bytes containing the full immediate value.

The individual ISA layout is highly dependent of the byte size (8, 9, 10 or 12 bits), the degree to which a full featured ISA is desired, and the number of codes in each instruction field. Combining instruction fields is possible, say, three data sizes and five addressing modes. Eliminating fields is also possible, for instance the return enable bit. In general, a two byte instruction will support an op-code and a stack reference. Three byte instructions are very useful for three operand instructions.

The intent of the architecture is to support high code density that is compatible with FPGA and RISC micro-architecture; including multiple instruction issue, pipelining and out of order instruction completion. It is expected that a FPGA implementation will use a multi-ported LUT RAM for the stack areas offering single clock per instruction execution. Most likely the TOS, the PC and the residue register (RR) will be virtual in the sense that a DFF implementation is present and virtually the TOS is part of the data stack, PC is part of the return stack and Residue is part of the global area.

[1] PDP-11 has R, a register reference, (R) a register indirect reference, (R++) a register indirect reference with post auto-increment, (--R) a register indirect reference with pre memory access data size decrement, (R+offset) a register indirect reference with a displacement field suffixed to the instruction and three other modes not considered here.

The RR captures signed and unsigned overflow, upper half of a multiply operation, remainder from a divide and round-off error from floating-point operations. Any instruction can access the residue. Three operand instructions are ideal for utilizing the RR.

At this point, issues of stack spill and refill, interrupt handling, memory caching and virtual memory management are not specified.