6.823 Computer System Architecture

EDSACjr-II

http://csg.csail.mit.edu/6.823/

The EDSACjr (Handout 1) requires using self-modifying code to implement common operations, which adds bookkeeping instructions. EDSACjr-II solves this problem by adding an *index register*, which is used in conjunction with the accumulator to simplify address calculations.

Table 1 summarizes the set of *additional* instructions that EDSACjr-II introduces over EDSACjr. These instructions allow using the index register to construct memory addresses, as well as directly manipulating the index register. The table uses the following notation:

- M[x] stands for the contents of the memory location addressed by x.
- Accum refers to the accumulator, and IX refers to the index register.
- ← signifies that data is transferred (copied) from the location to the right of the ← to the location on the left.
- The immediate variable n is an address or a literal depending on the context.

Note that these instructions are *in addition* to the original set of instructions for EDSACjr – i.e. EDSACjr-II can still perform instructions that do not involve the index register such as ADD *n*, SUB *n*, etc.

Opcode	**Description**
ADDIX *n*	Accum ← Accum + M[*n* + IX]
SUBIX *n*	Accum ← Accum - M[*n* + IX]
LOADIX *n*	Accum ← M[*n* + IX]
STOREIX *n*	M[*n* + IX] ← Accum
ADDi *n*	IX ← IX + *n*
SUBi *n*	IX ← IX - *n*
LOADi *n*	IX ← M[*n*]
STOREi *n*	M[*n*] ← IX
BGEi *n*	If IX ≥ 0 then PC ← *n*
BLTi *n*	If IX < 0 then PC ← *n*
BEQi *n*	If IX == 0 then PC ← *n*
END	Halt machine

Table 1. Additional instructions that EDSACjr-II introduces over EDSACjr.