|  |  |  |
| --- | --- | --- |
| IAS Instruction Set | | |
| **Instruction name** | **Opcode** | **Description** |
| S(x)->Ac+ | 1 | copy the number in Selectron location x into AC |
| S(x)->Ac- | 2 | same as #1 but copy the negative of the number |
| S(x)->AcM | 3 | same as #1 but copy the absolute value |
| S(x)->Ac-M | 4 | same as #1 but subtract the absolute value |
| S(x)->Ah+ | 5 | add the number in Selectron location x into AC |
| S(x)->Ah- | 6 | subtract the number in Selectron location x from AC |
| S(X)->AhM | 7 | same as #5, but add the absolute value |
| S(X)->Ah-M | 8 | same as #7, but subtract the absolute value |
| S(x)->R | 9 | copy the number in Selectron location x into AR |
| R->A | 10 | copy the number in AR to AC |
| S(x)\*R->A | 11 | Multiply the number in Selectron location x by the number in AR. Place the left half of the result in AC and the right half in AR. |
| A/S(x)->R | 12 | Divide the number in AC by the number in Selectron location x. Place the quotient in AR and the remainder in AC. |
| Cu->S(x) | 13 | Continue execution at the left-hand instruction of the pair at Selectron location x |
| Cu`->S(x) | 14 | Continue execution at the right-hand instruction of the pair at Selectron location x |
| Cc->S(x) | 15 | If the number in AC is >= 0, continue as in #13. Otherwise, continue normally. |
| Cc`->S(x) | 16 | If the number in AC is >= 0, continue as in #14. Otherwise, continue normally. |
| At->S(x) | 17 | Copy the number in AC to Selectron location x |
| Ap->S(x) | 18 | Replace the right-hand 12 bits of the left-hand instruction at Selectron location x by the right-hand 12 bits of the AC |
| Ap`->S(x) | 19 | Same as #18 but modifies the right-hand instruction |
| L | 20 | Shift the number in AC to the left 1 bit (new bit on the right is 0) |
| R | 21 | Shift the number in AC to the right 1 bit (leftmost bit is copied) |
| halt | 0 | Halt the program (see paragraph 6.8.5 of the IAS report) |

loop: S(x)->Ac+ n ;load n into AC

Cc->S(x) pos ;if AC >= 0, jump to pos

halt ;otherwise done

.empty ;a 20-bit 0

pos: S(x)->Ah+ sum ;add n to the sum

At->S(x) sum ;put total back at sum

S(x)->Ac+ n ;load n into AC

S(x)->Ah- one ;decrement n

At->S(x) n ;store decremented n

Cu->S(x) loop ;go back and do it again

n: .data 5 ;will loop 6 times total

one: .data 1 ;constant for decrementing n

sum: .data 0 ;where the running/final total is kept