CPU Instruction Set

The 2 most significant bits of every instruction determine the addressing mode (AM) used, enabling 4 possible instruction formats.
*240 possible instructions, 57 taken, 183 left

AM (2b)	OPCODE + OPERANDS (14b)

Single register addressing

AM (2b)	BA(1b)	OP (6b)	COND (4b)	Rs (3b)

BA = 0

AM (2b)	BA(1b)	OP (2b)	COND (4b)	Rs(3b)	BIT (4b)

BA = 1

Double register addressing

AM (2b)	OP (4b)	COND (4b)	Rd (3b)	Rs (3b)

Triple register Addressing

AM (2b)	OP (1b)	COND (4b)	Rd (3b)	Rs1 (3b)	Rs2 (3b)

MULT operations.

Direct addressing and control operations

AM (2b)	OP (2b)	Address (12b)

AM (2b)	OP (2b)	OFF(1b)	COP (7b)	COND (4b)

OFF bit = 0

AM (2b)	OP (2b)	OFF(1b)	COP (4b)	COND (4b)	s (3b)

OFF bit = 1

The **Status Register** is updated every cycle and contains 8 ‘flags’ that provide information about the ALU output of the previous instruction. The COND field of each addressing mode determines which bit of the status register needs to be set for the current instruction to be executed.

SREG	Z	N	C	T	V	S	A	I
COND	0000	0001	0010	0011	0100	0101	0110	0111
ASSEMBLY CODE	ZS	NS	CS	TS	VS	SS	A	IS

Bit SET

SREG	Z	N	C	T	V	S	A	I
COND	1000	1001	1010	1011	1100	1101	1110	1111
ASSEMBLY CODE	ZC	NC	CC	TC	VC	SC	A	IC

Bit CLEAR

Z: equal to 0

N: negative

C: a copy of ALU carry out

T: Bit Storage

V: Two’s complement overflow

S: Sign flag = N XOR V

A: always high for unconditional operations * 1110 is an invalid code for COND.

I: Global Interrupt Enable (for implementing interrupts if we have time)

Single register addressing mode

We make the distinction between operations that access individual bits or not.

No individual bit access (BA = 0): * 51 more possibilities

AM (2b)	BA(1b)	OP (6b)	COND (4b)	Rs (3b)

- JMR Rs // set PC to the value held in Rs if COND is satisfied
- JMI Rs // set PC to Rs if Rs < 0
- JEQ Rs // set PC to Rs if Rs = 0
- CAR Rs // call subroutine in address held by Rs if COND is satisfied
  // store PC in address held by STREG, increment STREG, set PC to address held by Rs
- LSR Rs // logical right shift
- ASR Rs // arithmetic shift right
- INV Rs // invert Rs
- TWC Rs // convert to two's complement (invert, add 1)
- INC Rs // increment Rs
- DEC Rs // decrement Rs
- LDI Rd, N // direct load, 16-bit N always stored in next memory address **(3 cycles)**
- AIM Rd, N // ADD immediate (3 cycle)
- SIM Rd, N // SUB immediate (3 cycle)

Individual bit access (BA = 1): * 2 more possibilities

AM (2b)	BA(1b)	OP (2b)	COND (4b)	Rs(3b)	BIT (4b)

- SEB Rs, k // set bit k of Rs HIGH
- CLB RS, k // set bit k of Rs LOW
- STB Rs, k // store bit k of Rs in 'T' flag of SREG
- LOB Rs, k // load 't' flag of SREG onto bit k of Rs.

Double register addressing mode

AM (2b)	OP (4b)	COND (4b)	Rd (3b)

- ADD Rd, Rs // Rd = Rd + Rs;
- ADC Rd, Rs // Rd = Rd + Rs + C (carry flag)
- SUB Rd, Rs // Rd = Rd – Rs
- SBC // Rd = Rd – Rs – C
- GHA Rd, Rs // ghost ADD
  // same as ADD but Leave SREG intact
- GHS Rd, Rs // ghost SUB
  // same as SUB but Leave SREG intact
- MOV Rd, Rs // Rd = Rs
- MOW Rd, Rs // move word; (Rd+1):Rd = (Rs+1):Rs
- PUSH Rd, Rs // data in Rs gets stored in address held by Rd

  // like LOAD but Rd is post-incremented (for stack management)
- LOAD Rd, Rs // data in address held by Rs gets stored in Rd
- POP Rd, Rs // data in address held by Rs gets stored in Rd

// like STORE but Rd is pre-decremented (for stack management)

- STORE Rd, Rs // data in Rs gets stored in address held by Rd
- AND Rd, Rs // Rd = Rd • Rs (logical AND)
- OR Rd, Rs // Rd = Rd + Rs (logical OR)
- XOR Rd, Rs // Rd = Rd XOR Rs (logical exclusive OR)
- COMP Rd, Rs // do Rd – Rs and update SREG, don’t store result

Triple register addressing mode

AM (2b)	OP (1b)	COND (4b)	Rd (3b)	Rs1 (3b)	Rs2 (3b)

- MUL Rd, Rs1, Rs2 // (Rd+1):Rd = Rs1 * Rs2

  // result goes to register pair where Rd contains 16 LSB.
- MLS Rd, Rs1, Rs2 // same as MUL for all values in two’s complement

Direct addressing and control operations

AM (2b)	OP (2b)	Address (12b)

- **(00)** JMD N // Set PC to N
- **(01)** CALL N // save contents of program counter to address contained in the stack register
  // increment stack register
  // set program counter to N
  // unconditional
- **(10)** LDA N // load 12-bit address on R0 (by default)
- **(11)** denotes **control operations** that don't require addressing.

AM (2b)	OP (2b)	COP (5b)	COND (4b)

COND (0000…1111): denotes each bit of the SREG, and whether it's SET or CLEAR

Using the OFF bit, we then distinguish between instructions that require an offset or not.

AM (2b)	OP (2b)	OFF(1b)	COP (7b)	COND (4b)

OFF = 0

* 111 more possibilities

- RTN // decrement stack register
  // set program counter to value contained in the address held by this register
- STP // halt CPU.
- CLEAR // set ALL registers to 0x0

These instructions SET/CLEAR the SREG bits:

- SEZ // set Z
- CLZ // clear Z
- SEN // set N
- CLN // clear N
- SEC // etc
- CLC
- SET
- CLT
- SEV
- CLV
- SES
- CLS
- SEI
- CLI

AM (2b)	OP (2b)	OFF(1b)	COP (4b)	COND (4b)	s (3b)

OFF = 1

* 14 more possibilities

- BRU s // branch up if COND is satisfied

  // increment PC by s (unsigned)

- BRD s // branch down if COND is satisfied
  // decrement PC by s (unsigned)