SAP-1 Specifications

Instructions:

Format:

- Single fixed length 8 bit instruction word -> **i i i i p p p p**
- Where:
  - **i** represents the instruction
  - **p** represents the instruction parameter

Composition:

- Each instruction will be made up of micro-instructions. These micro-instructions represent control lines within the CPU.
- There will be a maximum of 8 micro-steps (t_0 through t_7).

RAM:

- 4 bytes -> 4 bit address and 8 bit width.

RAM Access:

- Access to RAM will occur through the MAR.
- The MAR can only accesses the 4 LSB's of data on the bus.

General Purpose Registers:

- 2 general purpose registers exist: A and B.
- Each general purpose register is 8 bits.
- The 2 general purpose registers feed their values directly to the ALU.
- They will also be able to read/write values to the data bus.

ALU Operations:

- The ALU will have two operations, Add and Subtract.
- Subtraction is implemented by inverting register B and setting C_{in} high.
- Subtraction is always in the form of A-B.The ALU will constantly compute the addition or subtraction, depending on control inputs, of registers A and B regardless of use.

Flags Register:

- Flags persist until changed with a flag modifying instruction.
- There will be two flags generated by the ALU:
  - CF -> Carry Flag
  - ZF -> Zero Flag

SAP-1 Instruction Specification

Instructions Table Key:

X -> Bit ignored. Value has no affect on instructions execution.

A A A A -> Address bits. Address to a location in RAM.

I I I I -> Immediate bits. Value of an immediate to operate on.

Mnemonic	Instruction (i i i i)	Parameter (p p p p)	Description
NOP	0 0 0 0		CPU no operation

Mnemonic	Instruction (i i i i)	Parameter (p p p p)	Description
LDA	0 0 0 1	A A A A	Load value at address AAAA to Register A
ADD	0 0 1 0	A A A A	Add value at address AAAA to Register A
SUB	0 0 1 1	A A A A	Subtract value at address AAAA from Register A
STA	0 1 0 0	A A A A	Write Register A to address AAAA
LDI	0 1 0 1	I I I I	Load immediate IIII to Register A
JMP	0 1 1 0	A A A A	Jump to address AAAA
JC	0 1 1 1	A A A A	Jump to address AAAA if Carry
JZ	1 0 0 0	A A A A	Jump to address AAAA if Zero
	1 0 0 1		
	1 0 1 0		
	1 0 1 1		
	1 1 0 0		
CLR	1 1 0 1		Clear Output
OUT	1 1 1 0		Output Register A
HLT	1 1 1 1		Halt CPU

SAP-1 Micro-Code Specification

Microcode Bit Specification

The below table contains the mnemonic, position, and description of each micro-code control bit.

Bit Mnemonic	Bit Position	Bit Description
HLT	23	Halt
MI	22	MAR In
RI	21	RAM In
RO	20	RAM Out
II	19	Instruction Register In
IO	18	Instruction Register Out

Bit Mnemonic	Bit Position	Bit Description
AI	17	A Register In
AO	16	A Register Out
BI	15	B Register In
BO	14	B Register Out
ΣO	13	ALU Sum Out
SO	12	ALU Subtraction Out
FI	11	Flag Register In
OI	10	Output Register In
OC	9	Output Register Clear
O2	8	Output Register Set 2's Complement
CE	7	Counter Enable
CI	6	Counter In
CO	5	Counter Out
JC	4	Jump Carry
JZ	3	Jump Zero
	2	
	1	
NXT	0	Fetch Next Instruction

Note: The instruction fetch cycle takes up micro-steps t_0 and t_1 and is hardwired into the control unit. Therefore, all micro-code instructions below start on step t_2 (010).

Mnemonic	Instruction	Step (t_x)	Mnemonics	HLT | MI | RI | RO | II | IO | AI | AO | BI | BO | ΣO | SO | OI | OC | O2 | CE | CI | CO | XX | XX | XX | XX | XX | NXT
NOP	0 0 0 0	0 1 0	NXT	00000000000000000000001
LDA	0 0 0 1	0 1 0	IO | MI	01000100000000000000000
		0 1 1	RO | AI	00010010000000000000000
		1 0 0	NXT	00000000000000000000001
Add	0 0 1 0	0 1 0	IO | MI	01000100000000000000000
		0 1 1	RO | BI	00010000100000000000000
		1 0 0	EO | AI | FI	00000010001010000000000
		1 0 1	NXT	00000000000000000000001
SUB	0 0 1 1	0 1 0	IO | MI	01000100000000000000000
		0 1 1	RO | BI	00010000100000000000000

Mnemonic	Instruction	Step (t_x)	Mnemonics	HLT | MI | RI | RO | II | IO | AI | AO | BI | BO | ΣO | SO | OI | OC | O2 | CE | CI | CO | XX | XX | XX | XX | XX | NXT
		1 0 0	SO | EO | AI | FI	00000010001110000000000
		1 0 1	NXT	00000000000000000000001
STA	0 1 0 0	0 1 0	IO | MI	01000100000000000000000
		0 1 1	AO | RI	00100001000000000000000
		1 0 0	NXT	00000000000000000000001
LDI	0 1 0 1	0 1 0	IO | AI	00000110000000000000000
		0 1 1	NXT	00000000000000000000001
JMP	0 1 1 0	0 1 0	IO | CI	00000100000000001000000
		0 1 1	NXT	00000000000000000000001
JC	0 1 1 1	0 1 0	JC	00000000000000000010000
		0 1 1	NXT	00000000000000000000001
JZ	1 0 0 0	0 1 0	JZ	00000000000000000001000
		0 1 1	NXT	00000000000000000000001
CLR	1 1 0 1	0 1 0	OC	00000000000000010000000
		0 1 1	NXT	00000000000000000000001
OUT	1 1 1 0	0 1 0	AO | OI	00000001000001000000000
		0 1 1	NXT	00000000000000000000001
HLT	1 1 1 1	0 1 0	HLT	10000000000000000000000
		0 1 1	NXT	00000000000000000000001

SAP-1 Instruction Documentation

This section will go into more depth on each instruction. It will be less on machine implementation but more on what each instruction does in a more verbose manner.

NOP -> *No Operation*
- Clock Cycles: 2
- Sets Flags: None
- Parameters: None
- This instruction does nothing. Looking at the microcode implementation, immediately after fetching this instruction the NXT micro-op is loaded and the fetch of the next instruction starts.

LDA -> *Load Address to A*
- Clock Cycles: 4
- Sets Flags: None

- Parameters: 1
  - A A A A -> 4 bit address in RAM
- This instruction takes an address in RAM, and then loads the value at that address into register A.

ADD -> *Add B to A*
- Clock Cycles: 5
- Sets Flags: CF, ZF
- Parameters: 1
  - A A A A -> 4 bit address in RAM
- This instruction takes an address in RAM, and then loads the value at that address into register B. Then, register A and B are added together and the result is stored back into register A.

SUB -> *Subtract B from A*
- Clock Cycles: 5
- Sets Flags: CF, ZF
- Parameters: 1
  - A A A A -> 4 bit address in RAM
- This instruction takes an address in RAM, and then loads the value at that address into register B. Then, register B is subtracted from A and the result is stored back into register A.

STA -> *Store A*
- Clock Cycles: 4
- Sets Flags: None
- Parameters: 1
  - A A A A -> 4 bit address in RAM
- This instruction takes an address in RAM, and then writes the contents in register A to that address in RAM.

LDI -> *Load Immediate to A*
- Clock Cycles: 3
- Sets Flags: None
- Parameters: 1
  - i i i i -> 4 bit immediate value
- This instruction takes a 4 bit immediate value and writes it to register A.

JMP -> *Jump*
- Clock Cycles: 3
- Sets Flags: None
- Parameters: 1
  - A A A A -> 4 bit address in RAM
- This instruction takes an address to RAM, and then loads that address into the program counter. This allows you to execute instructions out of order. This instruction will always jump.

JC -> *Jump on Carry*
- Clock Cycles: 3
- Sets Flags: None

- Parameters: 1
  - A A A A -> 4 bit address in RAM
- This instruction takes an address to RAM, and then loads that address into the program counter if the CF is set in the FLAGS register. This allows you to optionally execute instructions out of order.

JZ -> *Jump on Zero*
- Clock Cycles: 3
- Sets Flags: None
- Parameters: 1
  - A A A A -> 4 bit address in RAM
- This instruction takes an address to RAM, and then loads that address into the program counter if the ZF is set in the FLAGS register. This allows you to optionally execute instructions out of order.

CLR -> *Clear Output*
- Clock Cycles: 3
- Sets Flags: None
- Parameters: None
- This instruction clears the output display setting it back to 0.

OUT -> *Output Register A*
- Clock Cycles: 3
- Sets Flags: None
- Parameters: None
- This instruction displays the contents of register A on the output display.

HLT -> *Halt*
- Clock Cycles: 3
- Sets Flags: None
- Parameters: None
- This instruction freezes the CPU until it is reset. This command effectively disconnects the clock signal.