Dominik Salvet, 2016

Six Alpha
Programmer's manual

Table of Contents

1 A brief description of Six Alpha

Six Alpha is processor architecture with following characteristics:

- 4-bit width,
- Harvard type,
- RISC type,
- can address up to 16 nibbles data memory,
- can address up to 128 B instruction memory.

2 Memory model

For data and instruction memory is used physically addressed flat model - every address used in instruction is equal address that will appear on the address bus. However not all data address space is for RWM. If it's required to communicate with I/O devices, it is realized by eight ports which can be driven from control registers that are mapped in data address space.

2.1 Data memory layout

First 12 addresses are used for RWM, the rest of address space represent special purpose registers:

- **SCR** - Status and Control Register
    - CF - Carry Flag
        - the result of ALU is out of range
    - ZF - Zero Flag
        - the result of ALU is zero
- **POUT** - Port OUT
    - if PADDR is configured in write mode, then value in this register will be written on port
- **PIN** - Port IN
    - if PADDR is configured in read mode, then value from port can be read from this register, write to this register will be ignored
- **PADDR** - Port ADDRess
    - defines port operation mode and port address
    - W - write/!read mode*
    - addr3 - port address

addr	**description**	**internal structure**				**reset value**			
0 - 11	RWM								
12	SCR	CF	ZF			0	0	0	0
13	POUT								
14	PIN								
15	PADDR	W	addr3			0	0	0	0

* Write is performed every EXEC phase until W bit is cleared.

3 Register file

Architecture defines one main register, known as accumulator, which is used in every instruction except jump. Also there is IP register that holds address of actual executed instruction.

4 List of instructions

- JMP - jump
- LI - load immediate
- LD - load
- ST - store
- ADD - addition
- SUB - subtraction
- XOR - exclusive logical disjunction
- NAND - negation of logical conjunction
- SB - skip if bit
- SNB - skip if not bit
- SETB - set bit
- CLRB - clear bit

5 Instruction types

Instruction type is defined by number of zeros in higher part of instruction word.

5.1 Introducing

The following concepts `addr7`, `data4`, `baddr` are representing any natural number in range:

- `addr7` - from 0 to 2^7 - 1,
- `data4` - from 0 to 2^4 - 1,
- `baddr` - from 0 to 2^2 - 1.

These numbers can be expressed in various radix:

- Decimal - <digits(0-9)> or <digits(0-9)>D or <digits(0-9)>d,
- Hexadecimal - <digits(0-F or 0-f)>H or <digits(0-F or 0-f)>h,
- Octal - <digits(0-7)>O or <digits(0-7)>o,
- Binary - <digits(0,1)>B or <digits(0,1)>b.

5.2 Jump

Jump to absolute 7-bit value that is contained inside an instruction word.

Instruction word structure

7	6	5	4	3	2	1	0
1	addr7						

List of instructions

mnemonic	C language
JMP addr7	IP = addr7;

5.3 Arithmetic, logic and memory

The first operand of these instructions is always A register and the second is pointer to memory with one exception, LI instruction where it is 4-bit constant.

Instruction word structure

7	6	5	4	3	2	1	0
0	type			data4			

List of instructions

type			mnemonic	C language
0	0	1	LI data4	A = data4;
0	1	0	LD data4	A = *data4;
0	1	1	ST data4	*data4 = A;
1	0	0	ADD data4	A += *data4;
1	0	1	SUB data4	A -= *data4;
1	1	0	XOR data4	A ^= *data4;
1	1	1	NAND data4	A = ~(A & *data4);

5.4 Bit

Each instruction works with only one bit. This bit is indexed like bit array with lowest two bits in the instruction word. This type includes the only mechanism to achieve conditional branching.

Instruction word structure

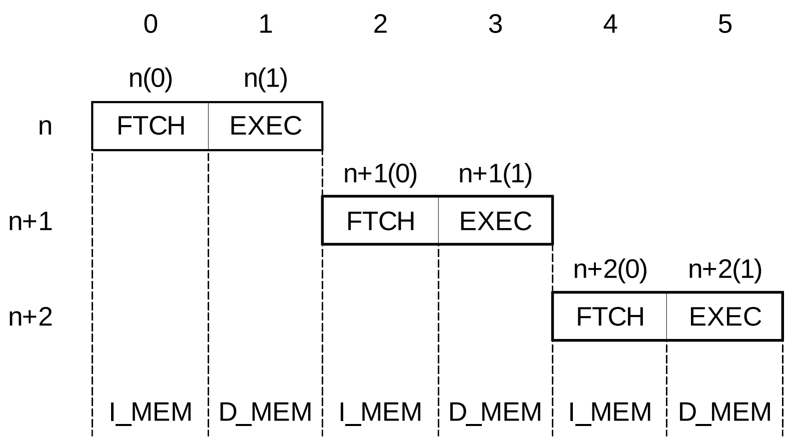
7	6	5	4	3	2	1	0
0	0	0	0	type		baddr	

List of instructions

type		mnemonic	C language
0	0	SB baddr	if(A[baddr] == 1) IP += 2;
0	1	SNB baddr	if(A[baddr] == 0) IP += 2;
1	0	SETB baddr	A[baddr] = 1;
1	1	CLRB baddr	A[baddr] = 0;

6 Instruction processing

Each instruction takes two rising edges of input clock signal and is not overlapped with another instruction.



6.1 Instruction phases

- FTCH - fetch instruction from instruction memory
- EXEC - execute instruction, access to data memory

7 Program example

```
; == Compare two numbers ==
; read port 3 and port 2 values and store their values
; subtract this values in the order port 2 - port 3
; write SCR register after this operation to port 1
; if port 2 value is greater than port 3, write ones to port 1
; then repeat all

;address  mnemonic  binary         comment

 00H:     LI 0      00010000
 01H:     ST 12     00111100       ; clear SCR

 02H:     LI 3      00010011       ; read port 3 command
 03H:     ST 15     00111111       ; store to paddr
 04H:     LD 14     00101110       ; load port 3 value from pin
 05H:     ST 7      00110111       ; store for later use

 06H:     LI 2      00010010       ; read port 2 command
 07H:     ST 15     00111111       ; store to paddr
 08H:     LD 14     00101110       ; load port 2 value from pin

 09H:     SUB 7     01010111       ; subtract store port 3 value
 0AH:     LD 12     00101100       ; load SCR
 0BH:     ST 13     00111101       ; store to pout
 0CH:     LI 9      00011001       ; write port 1 command
 0DH:     ST 15     00111111       ; store to paddr

 0EH:     LD 12     00101100       ; load SCR
 0FH:     SB 3      00000011       ; skip instruction if carry bit

 10H:     JMP 2     10000000       ; start again

 11H:     LI 15     00011111       ; load with ones
 12H:     ST 13     00111101       ; store to pout
 13H:     JMP 2     10000010       ; start again
```

The program is only for demonstrate the possibilities. Used algorithm is not intended to be optimized.

List of Symbols and Abbreviations

RWM	Read-write memory
IP	Instruction pointer
A	Accumulator