Specification

This document specifies the specifications for the design of a 16 bit computer.

Requirement Definitions

The following requirements are imposed on the design of the computer:

Clock	Variable from 0-1000 Hz
Stack Memory	Seperate stack memory to prevent stack overflows.
Registers	8 registers including a Stack Pointer Register and a Program Counter, additionally one general purpose shift register and 5 general purpose registers. The flag register has flags for negative, zero, overflow and carry
ALU	Addition, subtraction, AND, OR, NOT, left shifting, sign extension of bytes, flag register
Instruction Register	The first 5 bits define the instruction.

Architecture

The basic architecture is a Von-Neumann architecture where the memory and the CPU are connected by one single system bus.

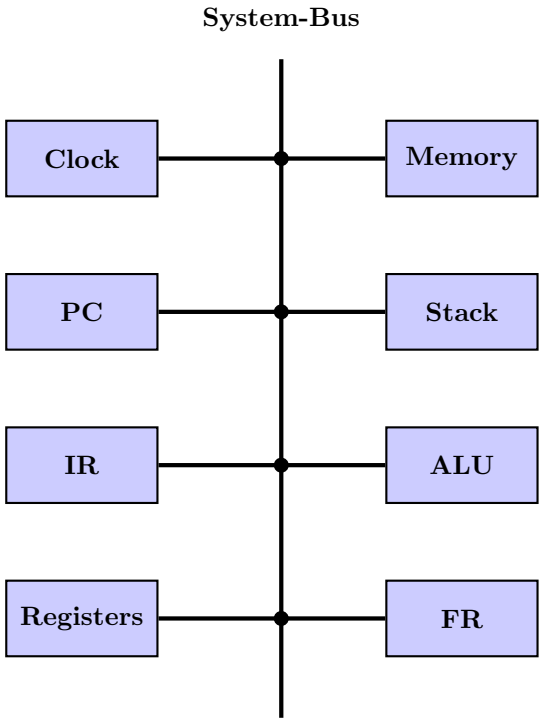


Figure 1: System architecture

Instruction Set

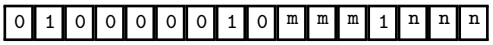
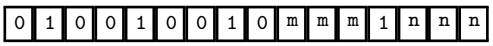
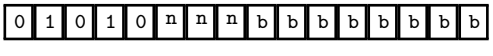
General Overview

The instruction set consists of 32 different instructions. The first 5 bit of the operation code are interpreted as the instruction. The instructions that have to implemented are the following:

Data transfer	1	`LDW`	Load a value of a memory address into a register
	2	`LDB`	Load the LSB of a memory address into a register
	3	`MOV`	Move a value from one register to another
	4	`MOV`	Move an immediate value in a register
	5	`STW`	Store the value of a register in a memory adress
	6	`STB`	Store the LSB of a register value in a memory address
Stack	7	`PSH`	Push register values onto the stack
	8	`POP`	Pop values from the stack
ALU	9	`ADD`	Add the values of two registers
	10	`ADD`	Add an immediate value to a register
	11	`SUB`	Subtract the values of two registers
	12	`SUB`	Subtract an immediate value from a register
	13	`MUL`	Multiply the values of two registers
	14	`AND`	Bitwise AND of the values of two registers
	15	`ORR`	Bitwise OR of the values of two registers
	16	`XOR`	Bitwise XOR of the values of two registers
	17	`NOT`	Bitwise inversion of the value in one register
	18	`LLS`	Shift the value in a register left by a variable amount
	19	`ALS`	Shift the value in a register arithmetically left by a variable amount
	20	`RLS`	Rotate the value in a register left by a variable amount
	21	`LRS`	Shift the value in a register right by a variable amount
	22	`ARS`	Shift the value in a register arithmetically right by a variable amount
	23	`RRS`	Rotate the value in a register right by a variable amount
	24	`SXT`	Sign extension of the LSB
Branching	25	`BRX`	Branch to the address stored in a register
	26	`BIF`	Branch if a certain flag is set in the flag register

The operation codes have the following general layout:

Instruction Set Architecture

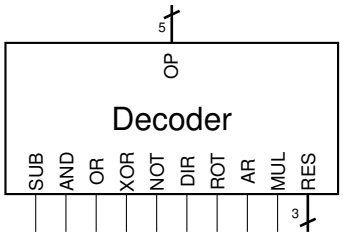
Operation Code	Mnemonic	Description
	LDW	Load the word that is stored at the address held by the register `Rmmm` into the register `Rnnn`.
	LDB	Load the least significant byte of the word at the address held by the register `Rmmm` into the register `Rnnn`.
	MOVE	Move the value `bbbb'bbbb` into the register `Rnnn`.

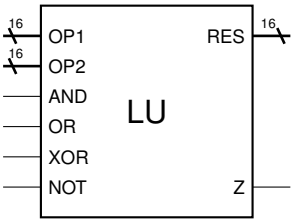
0 1 1 0 0 0 0 1 0 m m m 1 n n n	STRW	Store the value of the register `Rmmm` at the address held by the register `Rnnn`.
0 1 1 0 1 0 0 1 0 m m m 1 n n n	STRB	Store the least significant byte of the value in register `Rmmm` at the address held by the register `Rnnn`.
0 0 0 0 0 0 p s x x x x x x x x	PUSH	Push the declared registers `x`, the stack pointer `s` or the program counter `p` on the stack
0 0 0 0 1 0 p s x x x x x x x x	POP	Pop values from the stack in the declared registers `x`, the stack pointer `s` or the program counter `p`.
1 0 0 0 0 0 1 m m m n n n d d d	ADD	Add the values of the registers `Rmmm` and `Rnnn` and store the result in the register `Rddd`.
1 0 0 0 1 0 1 m m m n n n d d d	SUB	Subtract the value in the register `Rnnn` from the value in the register `Rmmm` and store the result in the register `Rddd`.
1 0 0 1 0 0 1 m m m n n n d d d	AND	Calculate a bitwise AND of the values in the registers `Rmmm` and `Rnnn`. Store the result in the register `Rddd`.
1 0 0 1 1 0 1 m m m n n n d d d	OR	Calculate a bitwise OR of the values in the registers `Rmmm` and `Rnnn`. Store the result in the register `Rddd`.
1 0 1 0 0 0 0 1 0 m m m 1 d d d	NOT	Bitwise inversion of the value in the register `Rmmm`. Store the result in the register `Rddd`.
1 0 1 0 1 0 0 1 0 m m m b b b b	LSHIFT	Shift the value in the register `Rmmm` by `bbbb` to the left.
1 0 1 1 0 0 0 1 0 m m m b b b b	RSHIFT	Shift the value in the register `Rmmm` by `bbbb` to the right.
1 0 1 1 1 0 0 0 0 0 0 0 1 n n n	SIXT	Signextend the least significant byte of the value that is stored in the register `Rnnn`.

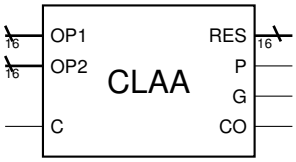
ALU

The ALU consists of the following components:

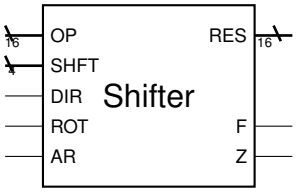
- carry lookahead adder (CLAA)
- barrel shifter
- wallace tree multiplier
- logic unit
- flag register


5
OP
Decoder
SUB
AND
OR
XOR
NOT
DIR
ROT
AR
MUL
RES
3


16
16
OP1
OP2
AND
OR
XOR
NOT
LU
RES
16
Z


16
16
OP1
OP2
C
CLAA
RES
16
P
G
CO


16
16
OP1
OP2
WTM
OUT1
16
OUT2
16


16
OP
4
SHFT
DIR
ROT
AR
Shifter
RES
16
F
Z