Fetch and Decode

CycleNr	Fetch&Decode
1	Connect PC to Mem Addr, IR to DBus
2	Do Mem Read and IR write Signals
3	Connect PC and 1(through multiplexer ctrl)to ALU, ADD ctrl
4	Reg write back

Memory Access

CycleNr	Read	Write
1	Apply Adress to inputs	Apply Adress and Data to inputs
2	Clear NotCS and NotOE	Set first NotOE and clear NotWE, then clear NotCS
3	wait > 6nS	wait > 6nS
4	Read Data from Databus	Set NotCS
5	Set NotCS	

CycleNr	ALU	MOV	LDI	LDA	STR	PUSH	POP	JMP	JMPZ	CALL	RET	
5	Connect Operand Reg and Akk to ALU-input Akk to ALU-output, apply ctrl signals	connects one register as read	read mem reg1=adress,	read mem reg1=address, reg2=data	write mem reg1=address,	connect SP, 1 to ALU, choose SUB,	read mem SP=address, reg2=data	same as MOV	connect PC and IR to DBus if(zero)	connect SP, 1 to ALU, choose SUB	connect SP to addrBus and PC to dBus	
6	latch the results	write Signal				write back	connect SP, 1 to ALU, choose ADD	do Signals	do signals	write back	do read signal	
7	drop latch out of tri state and write results into Akk					write mem sp=address, reg2=data,	write back			connetc PC to DBus and SP to addrBus	connect SP, 1 to ALU, choose ADD	
8										do read Signals	write back	
9												

MachineWord
One machine word consists of 16bits.
The first 4bits are the opcode bits/nibble which go directly into the instruction decoder. 4bits of the remaining 12bits are additional opcode bits but they are not used in every command. They indicate for example which registers to use... (output selector etc...)etc.
For the "mov" instruction, the remaining 12bits are used for immeadiate values.

Instructions

Name	Description	Nr Of Cycles	Example	Opcode	Implemented?	
NOR						
NAND					TRUE	
ADD	Add reg. Operand to Akkumulator, result is stored in Any register		ADD reg		TRUE	
SUB	Sub reg. Operand of Akkumulator, result is stored in Any register		SUB reg		TRUE	
MOV	Copy content from reg. to reg. Its The JMP as well. so just mov into pc		MOV destReg, sourceReg		TRUE	
LDI	Load immidiate operand into Akkumulator	6	LDI 183		TRUE	
LDA	Load memory pointed to by Operand register into sec. Operand Reg.		LDA destRegister, sourcePointerRegister		TRUE	
STR	Store sec. Operand Reg. where Operand Reg. points to		STR sourceReg, destPointerReg		TRUE	
PUSH	Push Operand Reg on Stack and inc. Operand Reg2. Note: an arbitrary register can be used a		PUSH reg, reg		TRUE	
POP	Pop from Stack into Operand		POP reg		FALSE	
JMPC	Jump to Operand if carry is set		JMPC 0xC3F		FALSE	
JMPZ	Jump to Operand if Akkumulator is zero		JMPZ 0xC3F		FALSE	
CALL	push return adress and jump to Operand		CALL 0xC3F		FALSE	
RET	Pop adress and jump to it		RET		FALSE	

Registers

Name	Application			
Akkumulator				
r1				
r2				
r3				
PC				
IR				
SP				

Modules - Description

Name		Description	