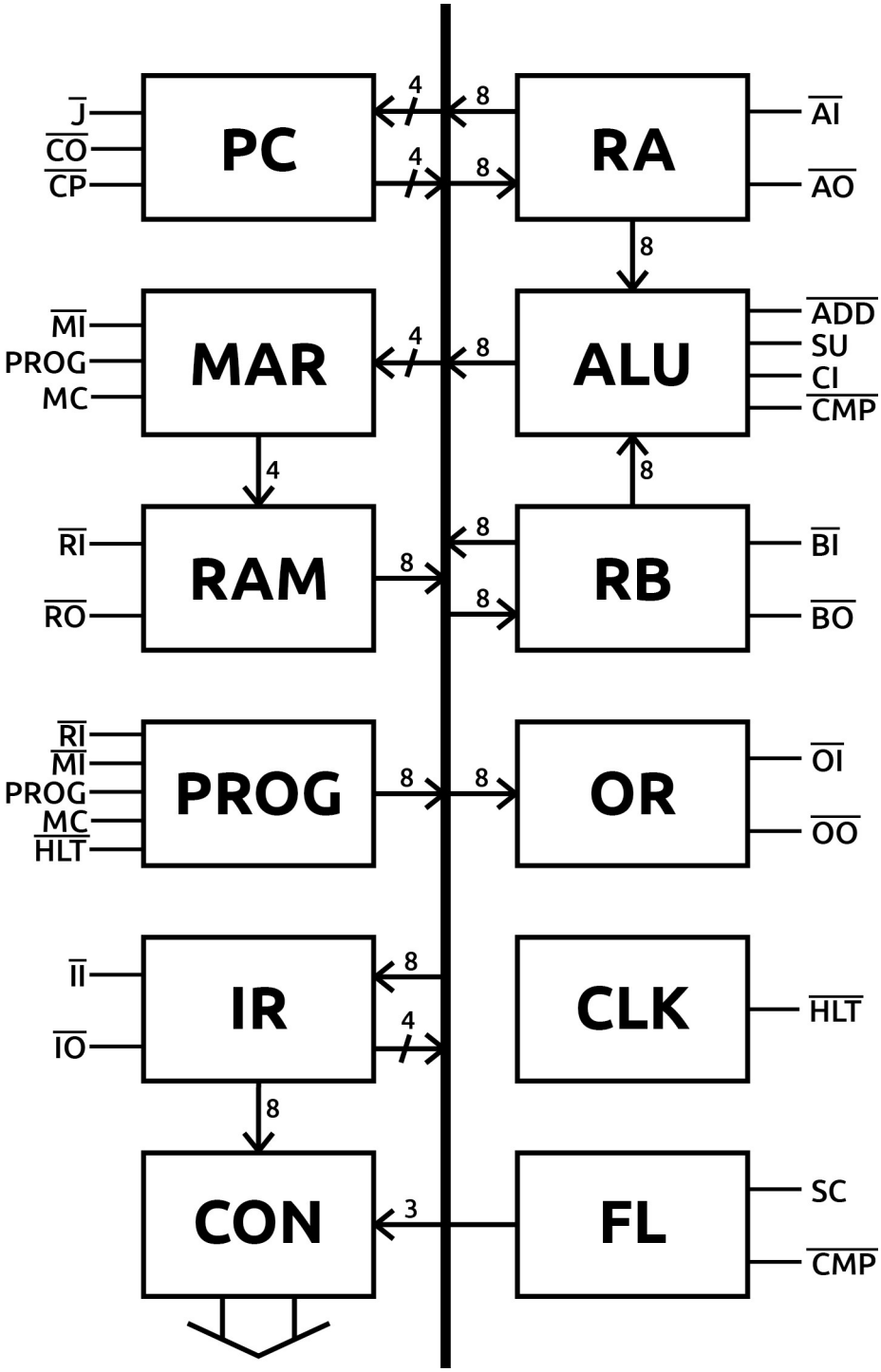
RazPU Mk I

- 8b data, 4b addresses
- 11 instructions
- Made from 74LS –family TTL logic chips
- Von Neumann architecture
- IEEE1284 (LPT) port programming interface (direct access to MAR, RAM, clock and few control signals)

Block diagram



Slashes denote use of lower half of the bus, numbers tell the width of the used bus. Over-line means active-when-low signal.

Control signals

/CO	Outputs contents of the program counter to the bus.
/CP	Rising edge increments the program counter by one.
/MI	Lower 4 bits of the bus are transferred to memory address register.
/RI	Contents of the bus are transferred to RAM to the address stored in MAR.
/RO	Contents of RAM at address stored in MAR is transferred to the bus.
/II	Contents of the bus are transferred to the instruction register.
/IO	Lower 4 bits of the instruction register [address of the operand] are transferred to the bus.
/AI	Contents of the bus are transferred to the A register.
/AO	Contents of the A register are transferred to the bus.
/BI	Contents of the bus are transferred to the B register.
/BO	Contents of the B register are transferred to the bus.
/OI	Contents of the bus are transferred to the output register.
/OO	Contents of the output register are transferred to... who knows where?
/ADD	The sum of registers A and B is transferred to the bus.
SU	Flips B register to turn addition to subtraction using two's complement.
CI	Enables carry in of the addition.
/CMP	Compares the contents of the registers A and B, sets flags (**EQ**, **LT**, and **GT**) accordingly.
SC	Updates the carry flag.
/J	Lower 4 bits of the bus are transferred to the program counter for a jump.
PROG	Activates **/HLT**, enables programming module.
MC	Clocks the MAR while programming.
/HLT	Stops the clock.

Instruction set

HLT

Halts the system.

MOV [src], [dest]

Move memory from source [src] to destination [dest]. Source and destination can be RA, RB, OR or RAM. However, moving memory from one register to another is not possible, i. e. one of the operands must always be RAM and the other must be a register. Moving memory from OR to RAM is not possible either.

The MOV instruction is implemented as LDA, STA, LDB, STB and OUT micro instructions.

Example:

MOV RA, 0xA ;move contents of the A register to RAM at address 0xA.

ADD

Store the sum of registers A and B to register A. Set carry flag [CF] according to the result.

RA = RA + RB

Example:

Suppose memory at 0xA contains decimal 4 and 0xB contains decimal 2.
MOV 0xA, RA ; RA = 4
MOV 0xB, RB ; RB = 2
ADD ; RA = 6

ADC
Store the sum of registers A and B and the carry flag [CF] to register A. Set carry flag [CF].

RA = RA + RB + CF

SUB
Store the difference of registers A and B to register A. Does not touch any flags.

RA = RA – RB

Example:

Suppose memory at 0xA contains decimal 4 and 0xB contains decimal 2.
MOV 0xA, RA ; RA = 4
MOV 0xB, RB ; RB = 2
SUB ; RA = 2

CMP
Compares RA to RB, and sets the flags [EQ, LT, GT].

Example:

Suppose memory at 0xA contains decimal 4 and 0xB contains decimal 2.
MOV 0xA, RA ; RA = 4
MOV 0xB, RB ; RB = 2
CMP ; RA>RB -> set [GT]

JMP [ADDR]
Unconditional jump to instruction specified at [ADDR]. Execution transfers to given address.

Example:

JMP 0x9 ; jump to address 0x9

JEQ [ADDR]
Jumps to instruction specified at [ADDR], if [EQ] flag is set by CMP.

Example:

Suppose memory at 0xA contains decimal 4 and 0xB contains decimal 2.
MOV 0xA, RA ; RA = 4
MOV 0xB, RB ; RB = 2
CMP ; RA>RB -> set [GT]
JEQ 0x9 ; doesn’t jump to address 0x9 because [EQ] flag is not set.

JNE [ADDR]
Jumps to instruction specified at [ADDR], if either or both [LT] or/and [GT] flags is/are set by CMP.

Example:

Suppose memory at 0xA contains decimal 4 and 0xB contains decimal 2.
MOV 0xA, RA ; RA = 4
MOV 0xB, RB ; RB = 2
CMP ; RA>RB -> set [GT]
JNE 9 ; jump to address 9 because [GT] flag is set.

JLT [ADDR]

Jumps to instruction specified at [ADDR], if [LT] flag is set by CMP.

Example:

Suppose memory at 0xA contains decimal 4 and 0xB contains decimal 2.
MOV 0xA, RA ; RA = 4
MOV 0xB, RB ; RB = 2
CMP ; RA>RB -> set [GT]
JLT 9 ; doesn't jump to address 9 because [LT] flag is not set.

JGT [ADDR]

Jumps to instruction specified at [ADDR], if [GT] flag is set by CMP.

Example:

Suppose memory at 0xA contains decimal 4 and 0xB contains decimal 2.
MOV 0xA, RA ; RA = 4
MOV 0xB, RB ; RB = 2
CMP ; RA>RB -> set [GT]
JGT 9 ; jump to address 9 because [GT] flag is set.

Opcodes

OPCODE			OPROM address								
FET			00	0	0	0	0	0	0	0	0
			01	0	0	0	0	0	0	0	1
			02	0	0	0	0	0	0	1	0
HLT	0	0 0 0 0	04	0	0	0	0	0	1	0	0
LDA	1	0 0 0 1	14	0	0	0	1	0	1	0	0
STA	2	0 0 1 0	24	0	0	1	0	0	1	0	0
LDB	3	0 0 1 1	34	0	0	1	1	0	1	0	0
STB	4	0 1 0 0	44	0	1	0	0	0	1	0	0
ADD	5	0 1 0 1	54	0	1	0	1	0	1	0	0
ADC	6	0 1 1 0	64	0	1	1	0	0	1	0	0
SUB	7	0 1 1 1	74	0	1	1	1	0	1	0	0
CMP	8	1 0 0 0	84	1	0	0	0	0	1	0	0
JMP	9	1 0 0 1	94	1	0	0	1	0	1	0	0
JEQ	A	1 0 1 0	A4	1	0	1	0	0	1	0	0
JNE	B	1 0 1 1	B4	1	0	1	1	0	1	0	0
JLT	C	1 1 0 0	C4	1	1	0	0	0	1	0	0
JGT	D	1 1 0 1	D4	1	1	0	1	0	1	0	0
OUT	E	1 1 1 0	E4	1	1	1	0	0	1	0	0

Microcode

SIG	/CO	/CP	/MI	/RI	/RO	/II	/IO	/AI	/AO	/BI	/BO	/OI	/ADD	SU	CI	/CMP	SC	J0	J1	J2	J3	/HLT	ACTIVE
BUS	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
FET	0	1	0	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	1	/CO, /MI
	1	0	1	1	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	1	/CP, /RO, /II
	1	1	0	1	1	1	0	1	1	1	1	1	1	0	0	1	0	0	0	0	0	1	/MI, /IO
HLT	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	/HLT
LDA	1	1	1	1	0	1	1	0	1	1	1	1	1	0	0	1	0	0	0	0	0	1	/RO, /AI
STA	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0	0	1	/RI, /AO
LDB	1	1	1	1	0	1	1	1	1	0	1	1	1	0	0	1	0	0	0	0	0	1	/RO, /BI
STB	1	1	1	0	1	1	1	1	1	1	0	1	1	0	0	1	0	0	0	0	0	1	/RI, /BO
ADD	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	1	/AI, /ADD, SC
ADC	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	0	0	0	0	1	/AI, /ADD, SC, CI
SUB	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	0	0	0	0	0	1	/AI, /ADD, SU
CMP	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	/CMP
JMP	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	0	1	0	0	0	1	/IO, J0
JEQ	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	0	0	0	0	1	1	/IO, J3
JNE	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	0	0	1	1	0	1	/IO, J1, J2
JLT	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	0	0	0	1	0	1	/IO, J2
JGT	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	0	0	1	0	0	1	/IO, J1
OUT	1	1	1	1	0	1	1	1	1	1	1	0	1	0	0	1	0	0	0	0	0	1	/RO, /OI