Utopium

Utopium

- MCU8051 processor
  - Several asynchronous examples
  - 256 instructions
    - Some reather complex
  - 256 bytes of ram with memory mapped:
    - Register bank + stack pointer
    - Accumulator, status flags
    - Bit addressable regions
- Started development on February 1st

8051 Instruction Set Table

	0000 .0	0001 .1	0010 .2	0011 .3	0100 .4	0101 .5	0110 .6	0111 .7	1000 .8	1001 .9	1010 .A	1011 .B	1100 .C	1101 .D	1110 .E	1111 .F	
							@R0	@R1	R0	R1	R2	R3	R4	R5	R6	R7	
0	0 | 1c | 1 NOP	1 | 2c | 2 AJMP 0A8	2 | 2c | 3 LJMP A16	3 | 1c | 1 RR A	4 | 1c | 1 INC A	5 | 1c | 1 INC A8	6 | 1c | 1 INC @R0	7 | 1c | 1 INC @R1	8 | 1c | 1 INC R0	9 | 1c | 1 INC R1	10 | 1c | 1 INC R2	11 | 1c | 1 INC R3	12 | 1c | 1 INC R4	13 | 1c | 1 INC R5	14 | 1c | 1 INC R6	15 | 1c | 1 INC R7	INC
1	16 | 2c | 3 JBC BIT,AR	17 | 2c | 2 ACALL 0A8	18 | 2c | 3 LCALL A16	19 | 1c | 1 RRC A	20 | 1c | 1 DEC A	21 | 1c | 1 DEC A8	22 | 1c | 1 DEC @R0	23 | 1c | 1 DEC @R1	24 | 1c | 1 DEC R0	25 | 1c | 1 DEC R1	26 | 1c | 1 DEC R2	27 | 1c | 1 DEC R3	28 | 1c | 1 DEC R4	29 | 1c | 1 DEC R5	30 | 1c | 1 DEC R6	31 | 1c | 1 DEC R7	DEC
2	32 | 2c | 3 JB BIT,AR	33 | 2c | 2 AJMP 1A8	34 | 2c | 1 RET	35 | 1c | 1 RL A	36 | 1c | 2 ADD A,#D	37 | 1c | 2 ADD A,A8	38 | 1c | 1 ADD A,@R0	39 | 1c | 1 ADD A,@R1	40 | 1c | 1 ADD A,R0	41 | 1c | 1 ADD A,R1	42 | 1c | 1 ADD A,R2	43 | 1c | 1 ADD A,R3	44 | 1c | 1 ADD A,R4	45 | 1c | 1 ADD A,R5	46 | 1c | 1 ADD A,R6	47 | 1c | 1 ADD A,R7	ADD
3	48 | 2c | 3 JNB BIT,AR	49 | 2c | 2 ACALL 1A8	50 | 2c | 1 RETI	51 | 1c | 1 RLC A	52 | 1c | 2 ADDC A,#D	53 | 1c | 2 ADDC A,A8	54 | 1c | 1 ADDC A,@R0	55 | 1c | 1 ADDC A,@R1	56 | 1c | 1 ADDC A,R0	57 | 1c | 1 ADDC A,R1	58 | 1c | 1 ADDC A,R2	59 | 1c | 1 ADDC A,R3	60 | 1c | 1 ADDC A,R4	61 | 1c | 1 ADDC A,R5	62 | 1c | 1 ADDC A,R6	63 | 1c | 1 ADDC A,R7	ADDC
4	64 | 2c | 2 JC AR	65 | 2c | 2 AJMP 2A8	66 | 1c | 2 ORL A8,A	67 | 1c | 3 ORL A8,#D	68 | 1c | 2 ORL A,#D	69 | 1c | 2 ORL A,A8	70 | 1c | 1 ORL A,@R0	71 | 1c | 1 ORL A,@R1	72 | 1c | 1 ORL A,R0	73 | 1c | 1 ORL A,R1	74 | 1c | 1 ORL A,R2	75 | 1c | 1 ORL A,R3	76 | 1c | 1 ORL A,R4	77 | 1c | 1 ORL A,R5	78 | 1c | 1 ORL A,R6	79 | 1c | 1 ORL A,R7	ORL
5	80 | 2c | 2 JNC AR	81 | 2c | 2 ACALL 2A8	82 | 1c | 2 ANL A8,A	83 | 1c | 3 ANL A8,#D	84 | 1c | 2 ANL A,#D	85 | 1c | 2 ANL A,A8	86 | 1c | 1 ANL A,@R0	87 | 1c | 1 ANL A,@R1	88 | 1c | 1 ANL A,R0	89 | 1c | 1 ANL A,R1	90 | 1c | 1 ANL A,R2	91 | 1c | 1 ANL A,R3	92 | 1c | 1 ANL A,R4	93 | 1c | 1 ANL A,R5	94 | 1c | 1 ANL A,R6	95 | 1c | 1 ANL A,R7	ANL
6	96 | 2c | 2 JZ AR	97 | 2c | 2 AJMP 3A8	98 | 1c | 2 XRL A8,A	99 | 1c | 3 XRL A8,#D	100 | 1c | 2 XRL A,#D	101 | 1c | 2 XRL A,A8	102 | 1c | 1 XRL A,@R0	103 | 1c | 1 XRL A,@R1	104 | 1c | 1 XRL A,R0	105 | 1c | 1 XRL A,R1	106 | 1c | 1 XRL A,R2	107 | 1c | 1 XRL A,R3	108 | 1c | 1 XRL A,R4	109 | 1c | 1 XRL A,R5	110 | 1c | 1 XRL A,R6	111 | 1c | 1 XRL A,R7	XRL
7	112 | 2c | 2 JNZ AR	113 | 2c | 2 ACALL 3A8	114 | 2c | 2 ORL C,BIT	115 | 2c | 1 JMP @A+DPTR	116 | 1c | 2 MOV A,#D	117 | 2c | 3 MOV A8,#D	118 | 2c | 2 MOV @R0,#D	119 | 2c | 2 MOV @R1,#D	120 | 1c | 2 MOV R0,#D	121 | 1c | 2 MOV R1,#D	122 | 1c | 2 MOV R2,#D	123 | 2c | 2 MOV R3,#D	124 | 1c | 2 MOV R4,#D	125 | 1c | 2 MOV R5,#D	126 | 1c | 2 MOV R6,#D	127 | 1c | 2 MOV R7,#D	MOV
8	128 | 2c | 2 SJMP AR	129 | 2c | 2 AJMP 4A8	130 | 2c | 2 ANL C,BIT	131 | 2c | 1 MOVC A,@A+PC	132 | 4c | 1 DIV AB	133 | 2c | 3 MOV A8,A8	134 | 2c | 2 MOV A8,@R0	135 | 2c | 2 MOV A8,@R1	136 | 2c | 2 MOV A8,R0	137 | 2c | 2 MOV A8,R1	138 | 2c | 2 MOV A8,R2	139 | 2c | 2 MOV A8,R3	140 | 2c | 2 MOV A8,R4	141 | 2c | 2 MOV A8,R5	142 | 2c | 2 MOV A8,R6	143 | 2c | 2 MOV A8,R7	MOV
9	144 | 2c | 3 MOV DPTR,#D16	145 | 2c | 2 ACALL 4A8	146 | 2c | 2 MOV BIT,C	147 | 2c | 1 MOVC A,@A+DPTR	148 | 1c | 2 SUBB A,#D	149 | 1c | 2 SUBB A,A8	150 | 1c | 1 SUBB A,@R0	151 | 1c | 1 SUBB A,@R1	152 | 1c | 1 SUBB A,R0	153 | 1c | 1 SUBB A,R1	154 | 1c | 1 SUBB A,R2	155 | 1c | 1 SUBB A,R3	156 | 1c | 1 SUBB A,R4	157 | 1c | 1 SUBB A,R5	158 | 1c | 1 SUBB A,R6	159 | 1c | 1 SUBB A,R7	SUBB
A	208 | 2c | 2 ORL C,/BIT	161 | 2c | 2 AJMP 5A8	162 | 1c | 2 MOV C,BIT	163 | 2c | 1 INC DPTR	164 | 4c | 1 MUL AB	165	166 | 2c | 2 MOV @R0,A8	167 | 2c | 2 MOV @R1,A8	168 | 2c | 2 MOV R0,A8	169 | 2c | 2 MOV R1,A8	170 | 2c | 2 MOV R2,A8	171 | 2c | 2 MOV R3,A8	172 | 2c | 2 MOV R4,A8	173 | 2c | 2 MOV R5,A8	174 | 2c | 2 MOV R6,A8	175 | 2c | 2 MOV R7,A8	MOV
B	176 | 2c | 2 ANL C,/BIT	177 | 2c | 2 ACALL 5A8	178 | 1c | 2 CPL BIT	179 | 1c | 1 CPL C	180 | 2c | 3 CJNE A,#D,AR	181 | 2c | 3 CJNE A,A8,AR	182 | 2c | 3 CJNE @R0,#D,AR	183 | 2c | 3 CJNE @R1,#D,AR	184 | 2c | 3 CJNE R0,#D,AR	185 | 2c | 3 CJNE R1,#D,AR	186 | 2c | 3 CJNE R2,#D,AR	187 | 2c | 3 CJNE R3,#D,AR	188 | 2c | 3 CJNE R4,#D,AR	189 | 2c | 3 CJNE R5,#D,AR	190 | 2c | 3 CJNE R6,#D,AR	191 | 2c | 3 CJNE R7,#D,AR	CJNE
C	192 | 2c | 2 PUSH A8	193 | 2c | 2 AJMP 6A8	194 | 1c | 2 CLR BIT	195 | 1c | 1 CLR C	196 | 1c | 1 SWAP A	197 | 1c | 2 XCH A,A8	198 | 1c | 1 XCH A,@R0	199 | 1c | 1 XCH A,@R1	200 | 1c | 1 XCH A,R0	201 | 1c | 1 XCH A,R1	202 | 1c | 1 XCH A,R2	203 | 1c | 1 XCH A,R3	204 | 1c | 1 XCH A,R4	205 | 1c | 1 XCH A,R5	206 | 1c | 1 XCH A,R6	207 | 1c | 1 XCH A,R7	XCH
D	208 | 2c | 2 POP A8	209 | 2c | 2 ACALL 6A8	210 | 1c | 2 SETB BIT	211 | 1c | 1 SETB C	212 | 1c | 1 DA A	213 | 2c | 3 DJNZ A8,AR	214 | 1c | 1 XCHD A,@R0	215 | 1c | 1 XCHD A,@R1	216 | 2c | 2 DJNZ R0,AR	217 | 2c | 2 DJNZ R1,AR	218 | 2c | 2 DJNZ R2,AR	219 | 2c | 2 DJNZ R3,AR	220 | 2c | 2 DJNZ R4,AR	221 | 2c | 2 DJNZ R5,AR	222 | 2c | 2 DJNZ R6,AR	223 | 2c | 2 DJNZ R7,AR	DJNZ
E	224 | 2c | 1 MOVX A,@DPTR	225 | 2c | 2 AJMP 7A8	226 | 2c | 1 MOVX A,@R0	227 | 2c | 1 MOVX A,@R1	228 | 1c | 1 CLR A	229 | 1c | 2 MOV A,A8	230 | 1c | 1 MOV A,@R0	231 | 1c | 1 MOV A,@R1	232 | 1c | 1 MOV A,R0	233 | 1c | 1 MOV A,R1	234 | 1c | 1 MOV A,R2	235 | 1c | 1 MOV A,R3	236 | 1c | 1 MOV A,R4	237 | 1c | 1 MOV A,R5	238 | 1c | 1 MOV A,R6	239 | 1c | 1 MOV A,R7	MOV
F	240 | 2c | 1 MOVX @DPTR,A	241 | 2c | 2 ACALL 7A8	242 | 2c | 1 MOVX @R0,A	243 | 2c | 1 MOVX @R1,A	244 | 1c | 1 CPL A	245 | 1c | 2 MOV A8,A	246 | 1c | 1 MOV @R0,A	247 | 1c | 1 MOV @R1,A	248 | 1c | 1 MOV R0,A	249 | 1c | 1 MOV R1,A	250 | 1c | 1 MOV R2,A	251 | 1c | 1 MOV R3,A	252 | 1c | 1 MOV R4,A	253 | 1c | 1 MOV R5,A	254 | 1c | 1 MOV R6,A	255 | 1c | 1 MOV R7,A	MOV

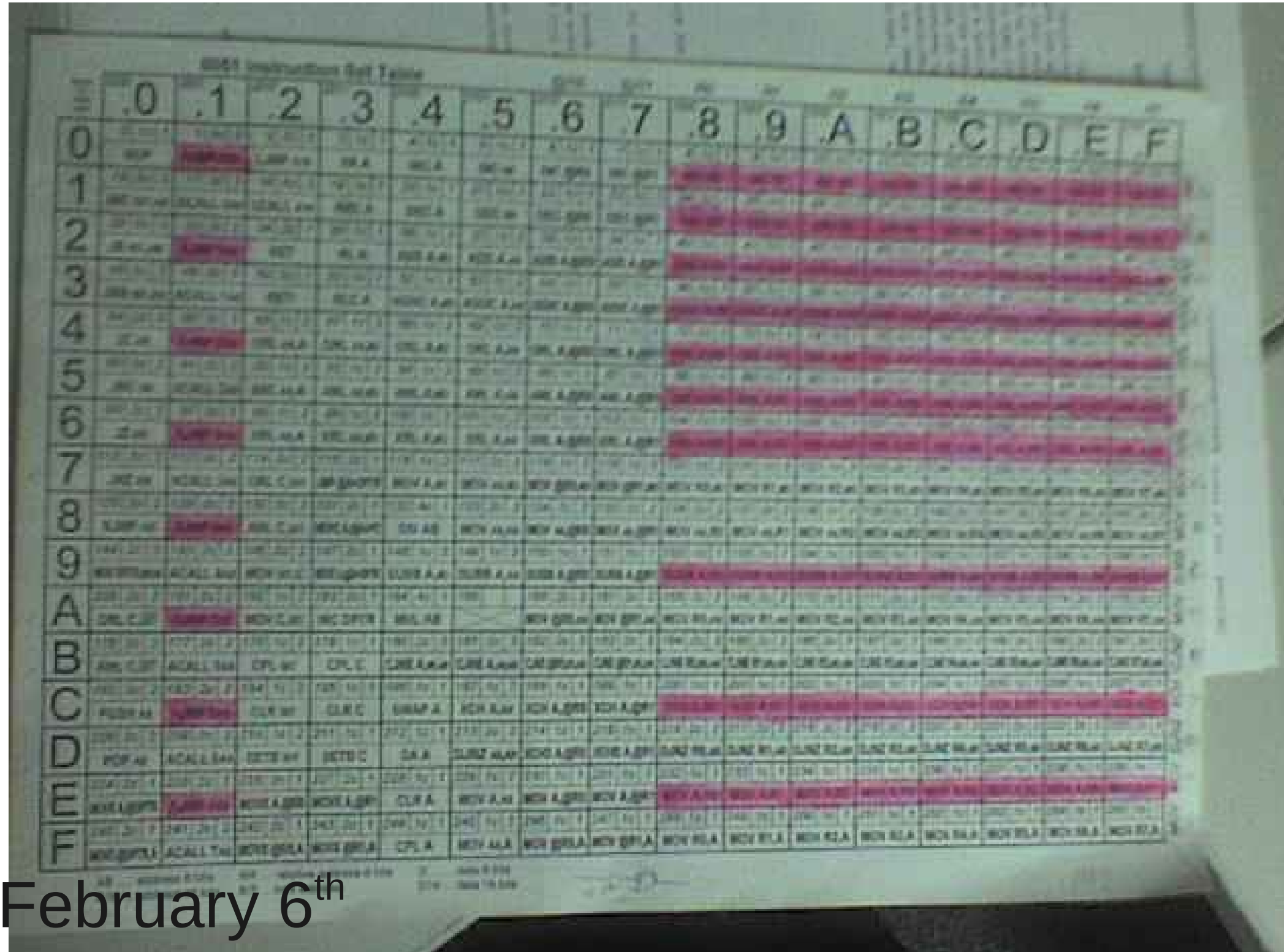
Matos (c) 1995, 2002

A8 ..... address 8 bits
A16 ... address 16 bits

AR .... relative address 8 bits
BIT ... bit's address

D ..... data 8 bits
D16 ... data 16 bits

Code | Cycles | Bytes
INSTRUCTION



February 6th

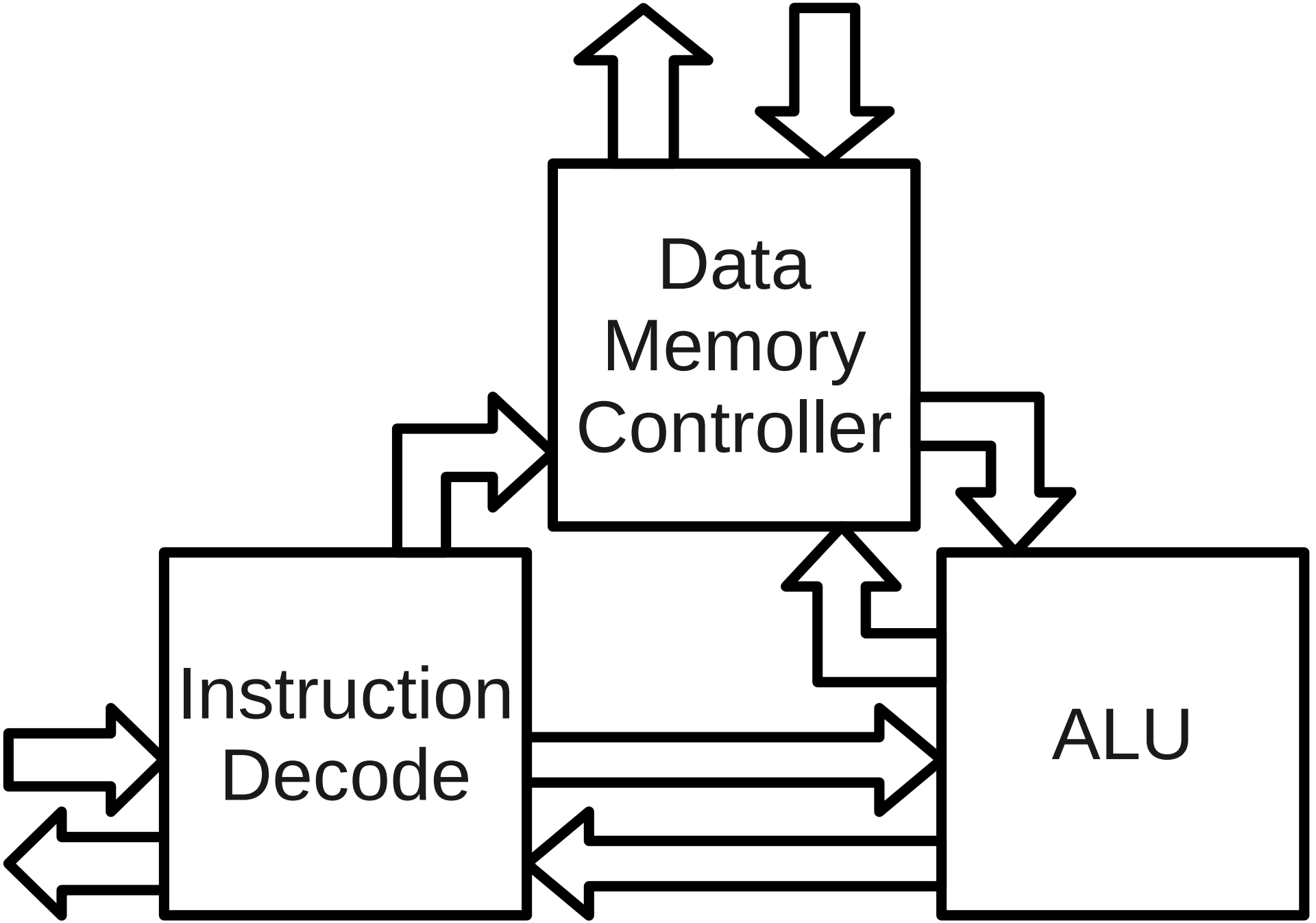
February 12th

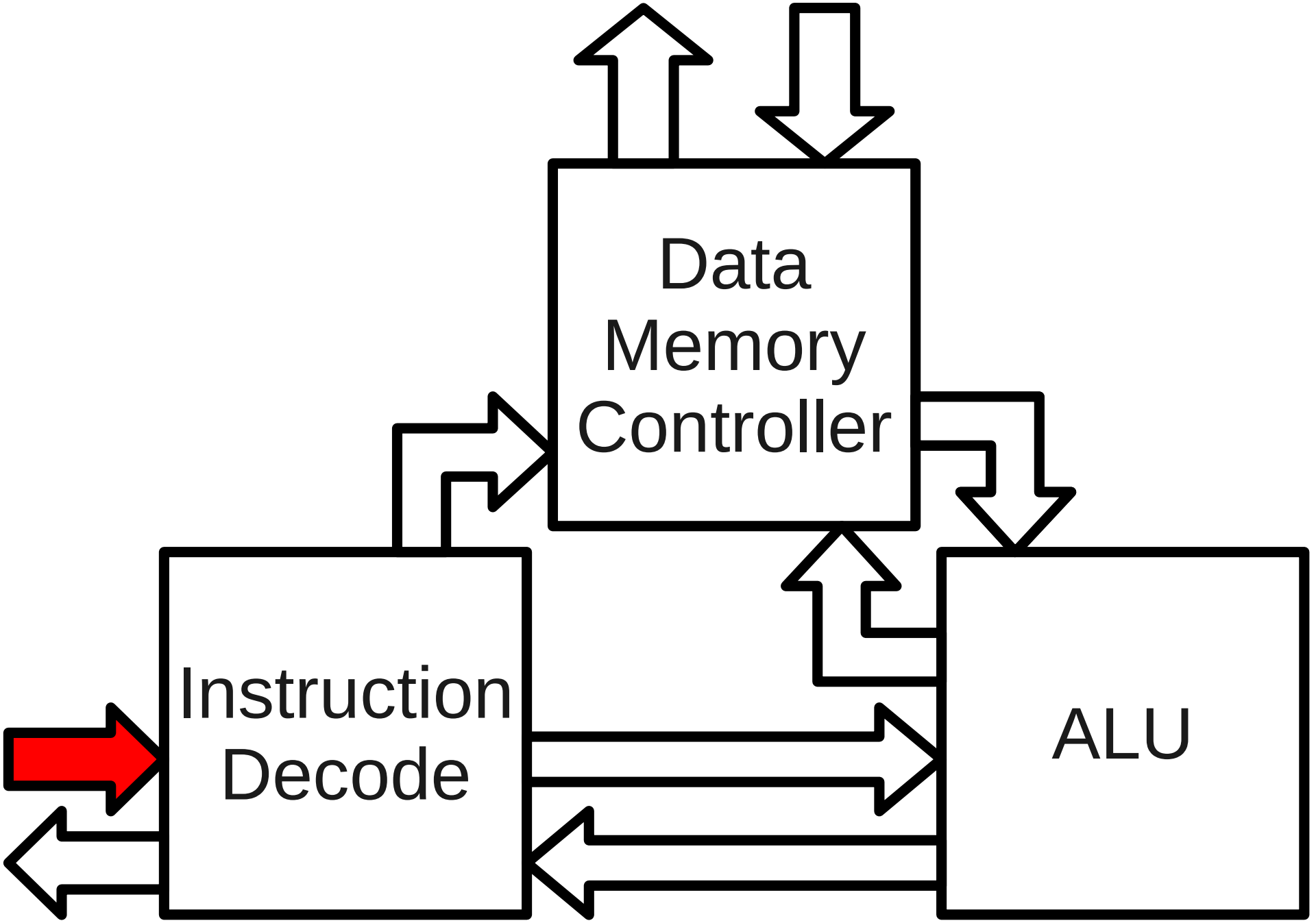
February 13th

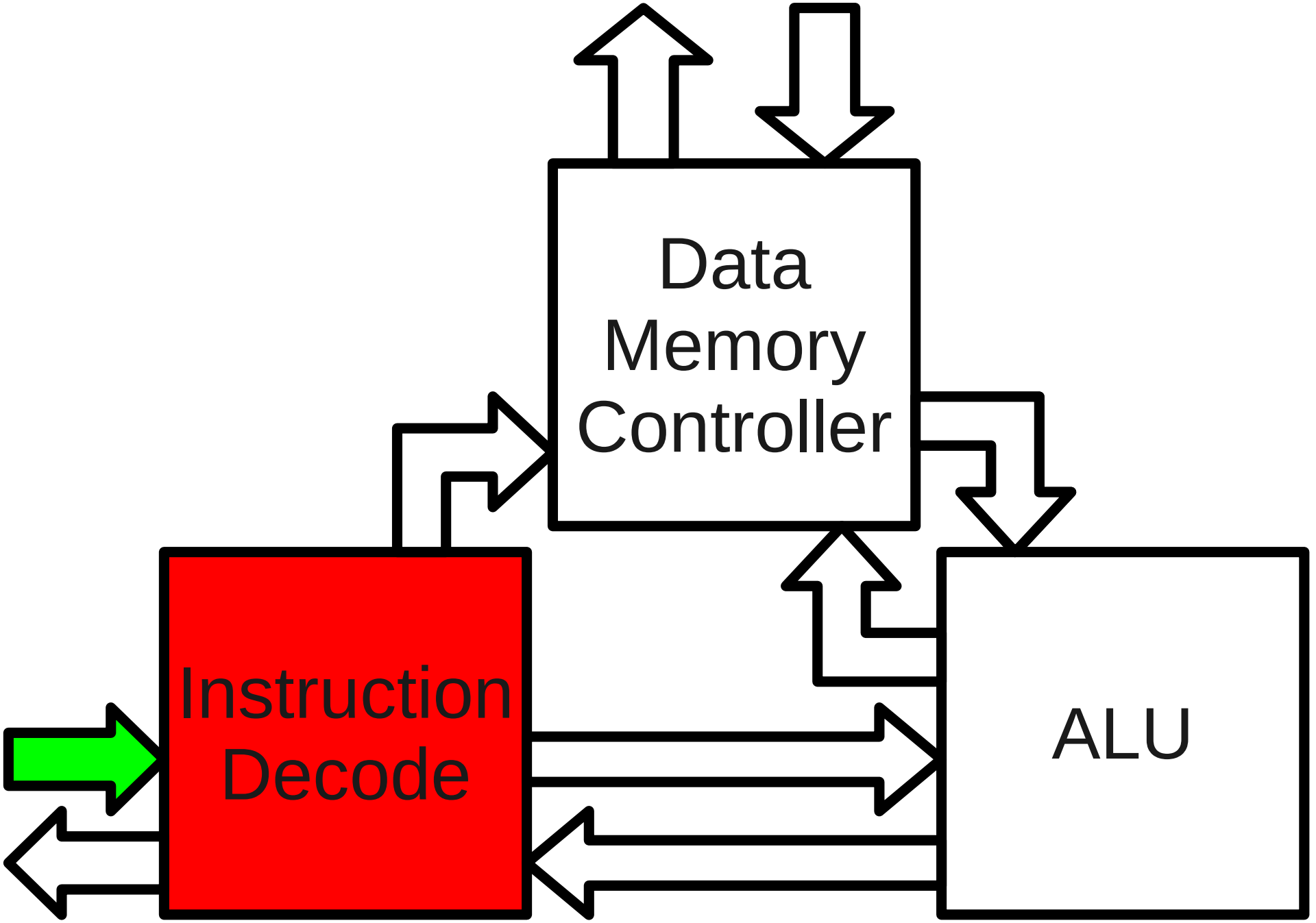
February 17th

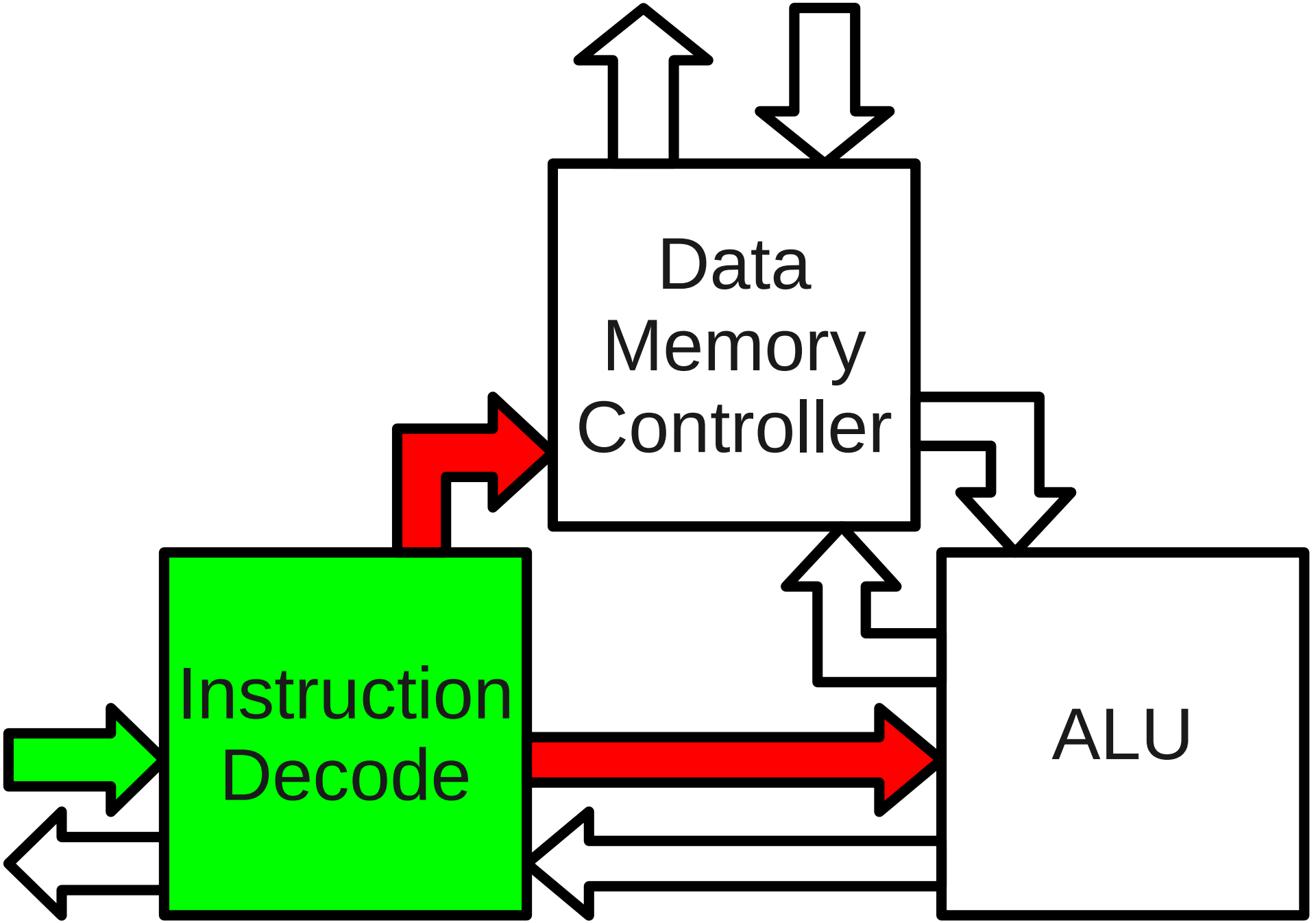
Functional development

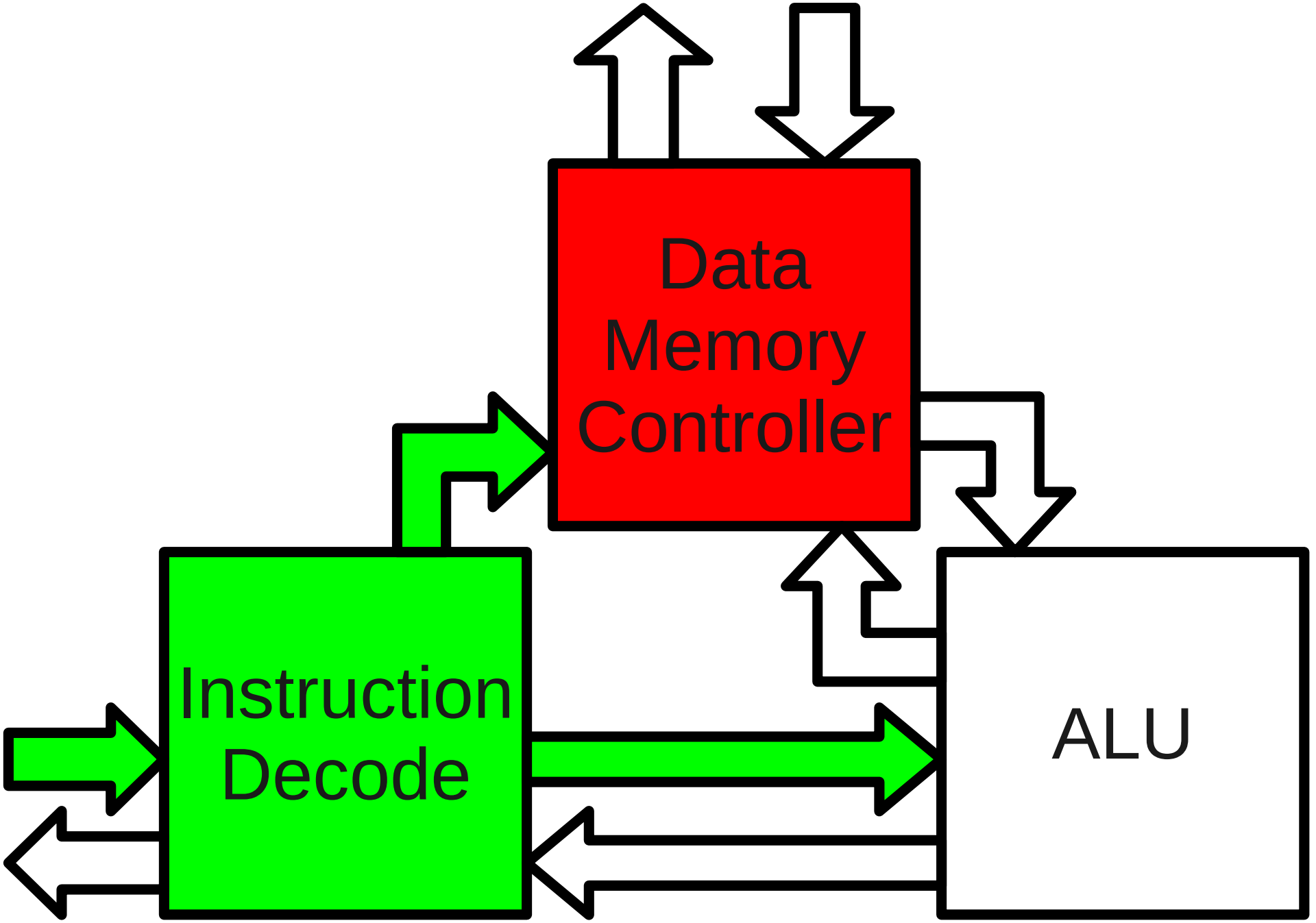
- 20 days to develop the synchronous version
- Desynchronised
  - Dual-rail early output
  - Wagging
- 10 days of tuning
  - Interesting stuff
- One man month processor
- 12 month of automating the process

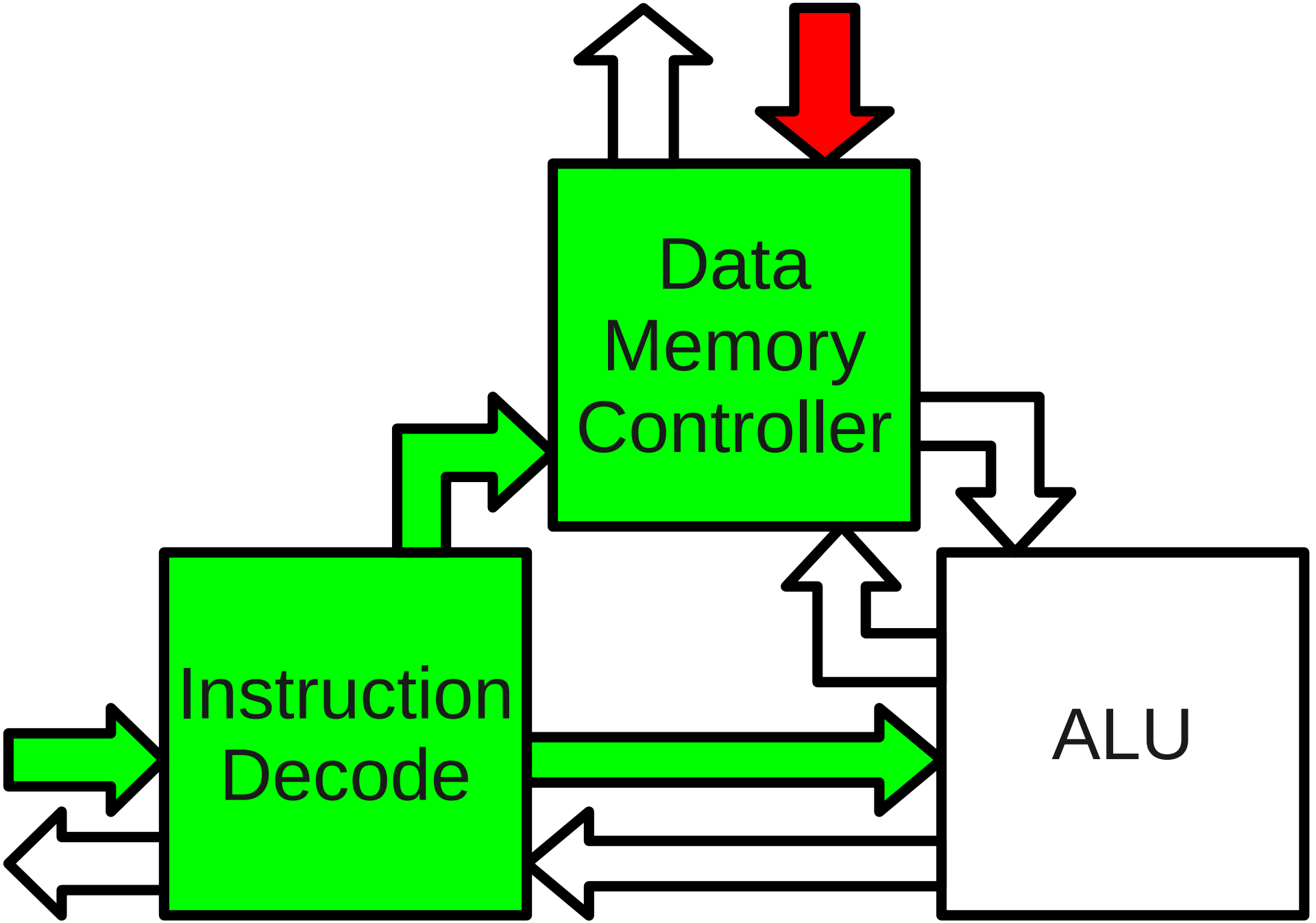

Data
Memory
Controller
Instruction
Decode
ALU

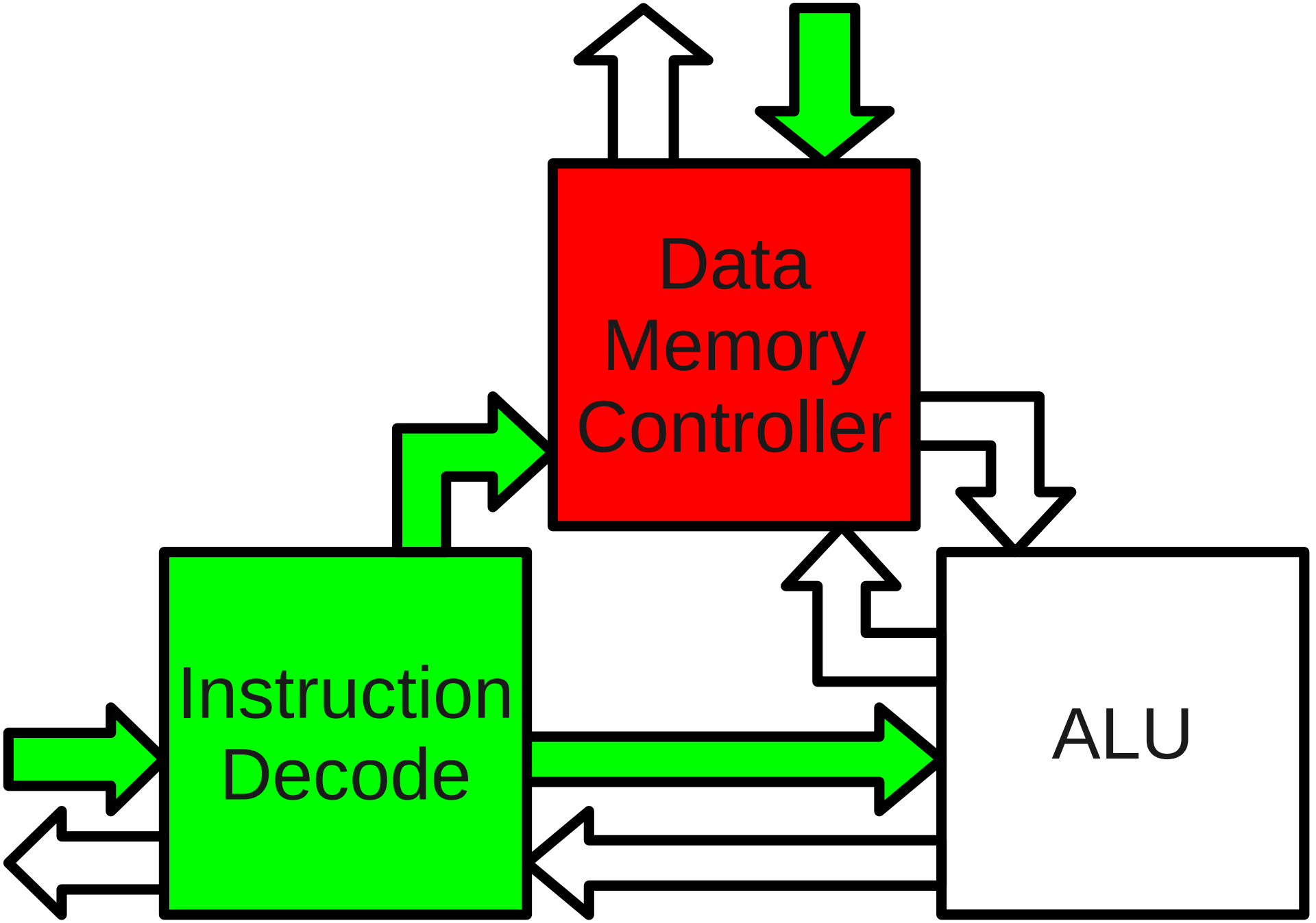

Data
Memory
Controller
Instruction
Decode
ALU

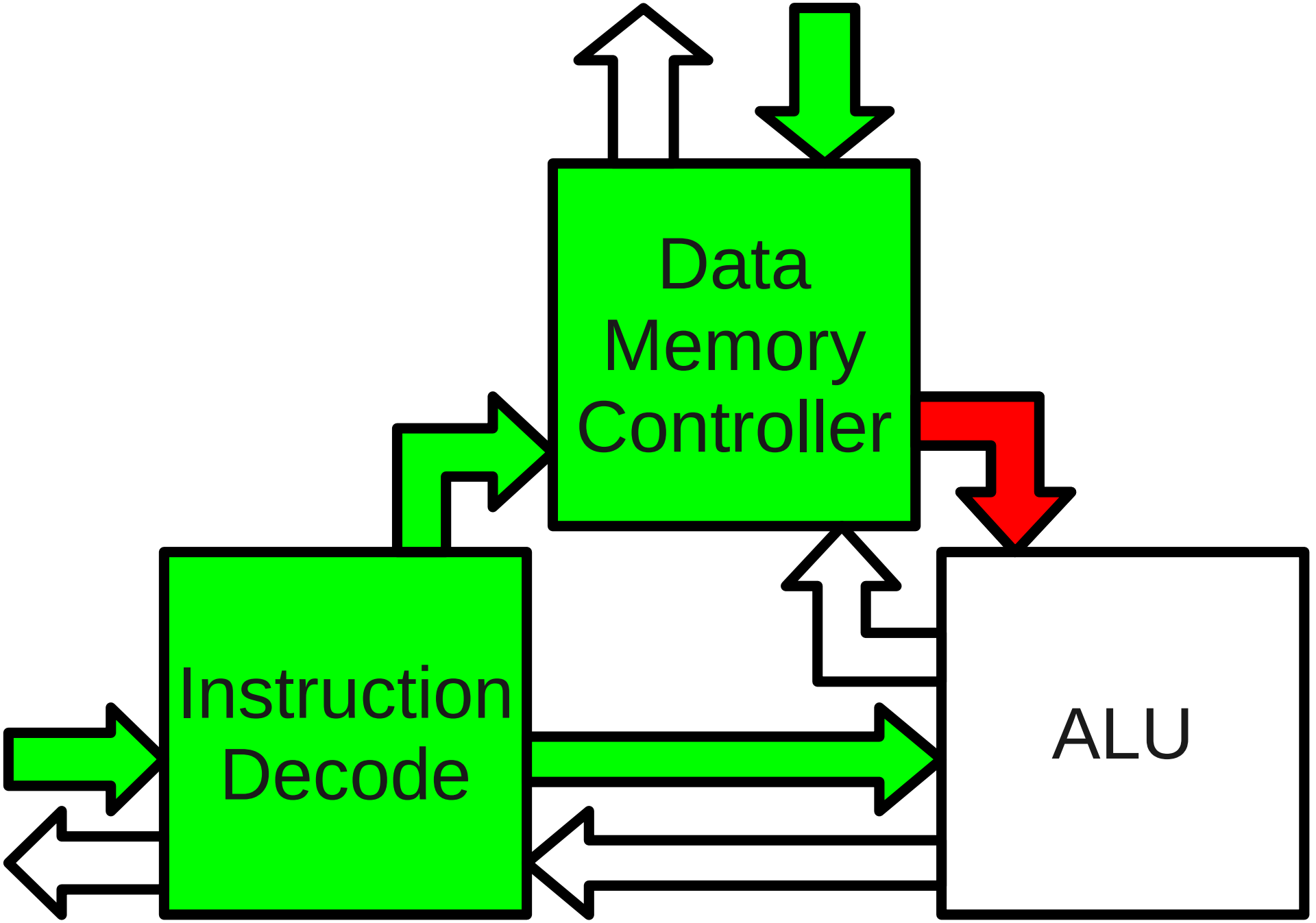

Data
Memory
Controller
Instruction
Decode
ALU

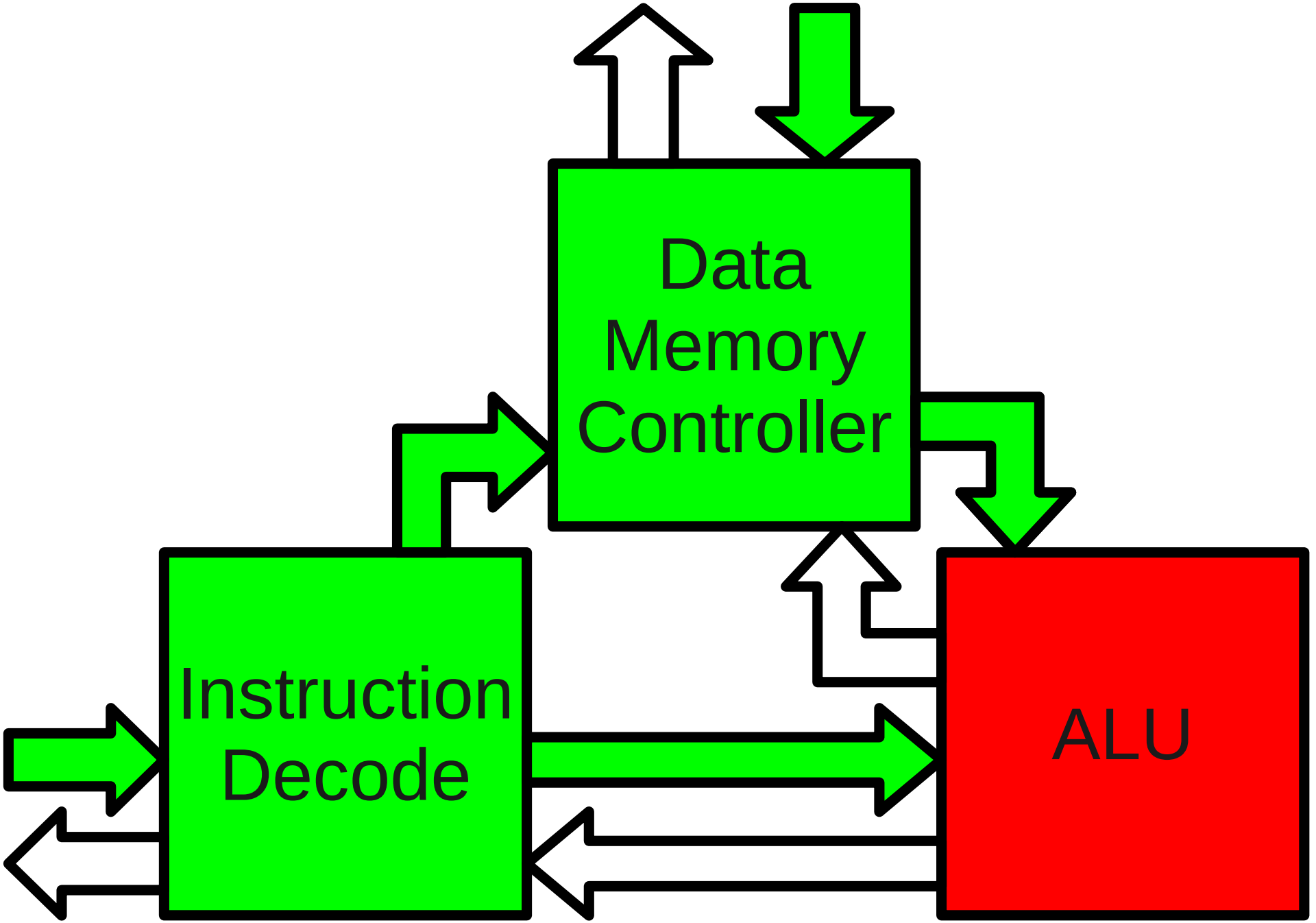

Data
Memory
Controller
Instruction
Decode
ALU

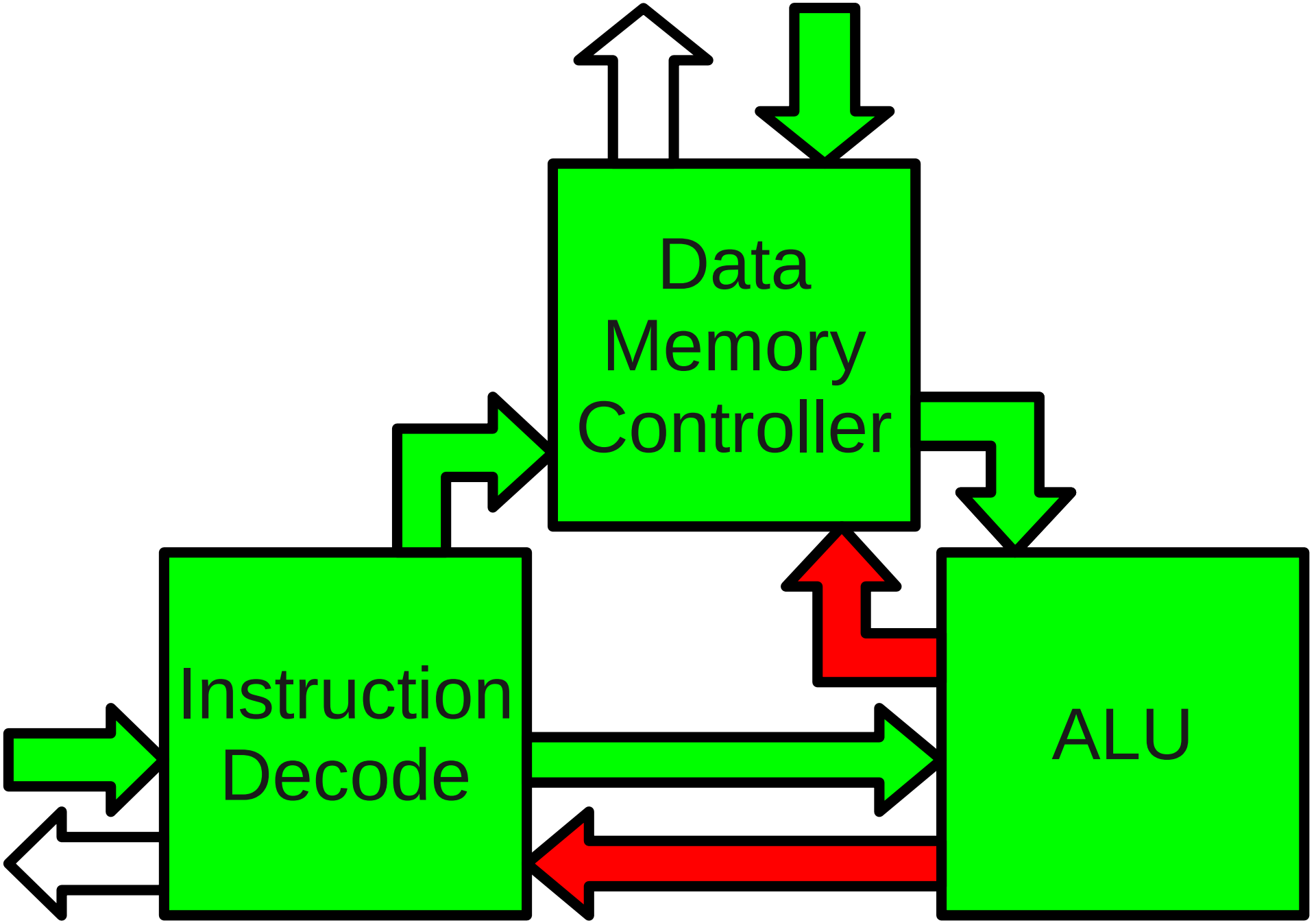

Data
Memory
Controller
Instruction
Decode
ALU

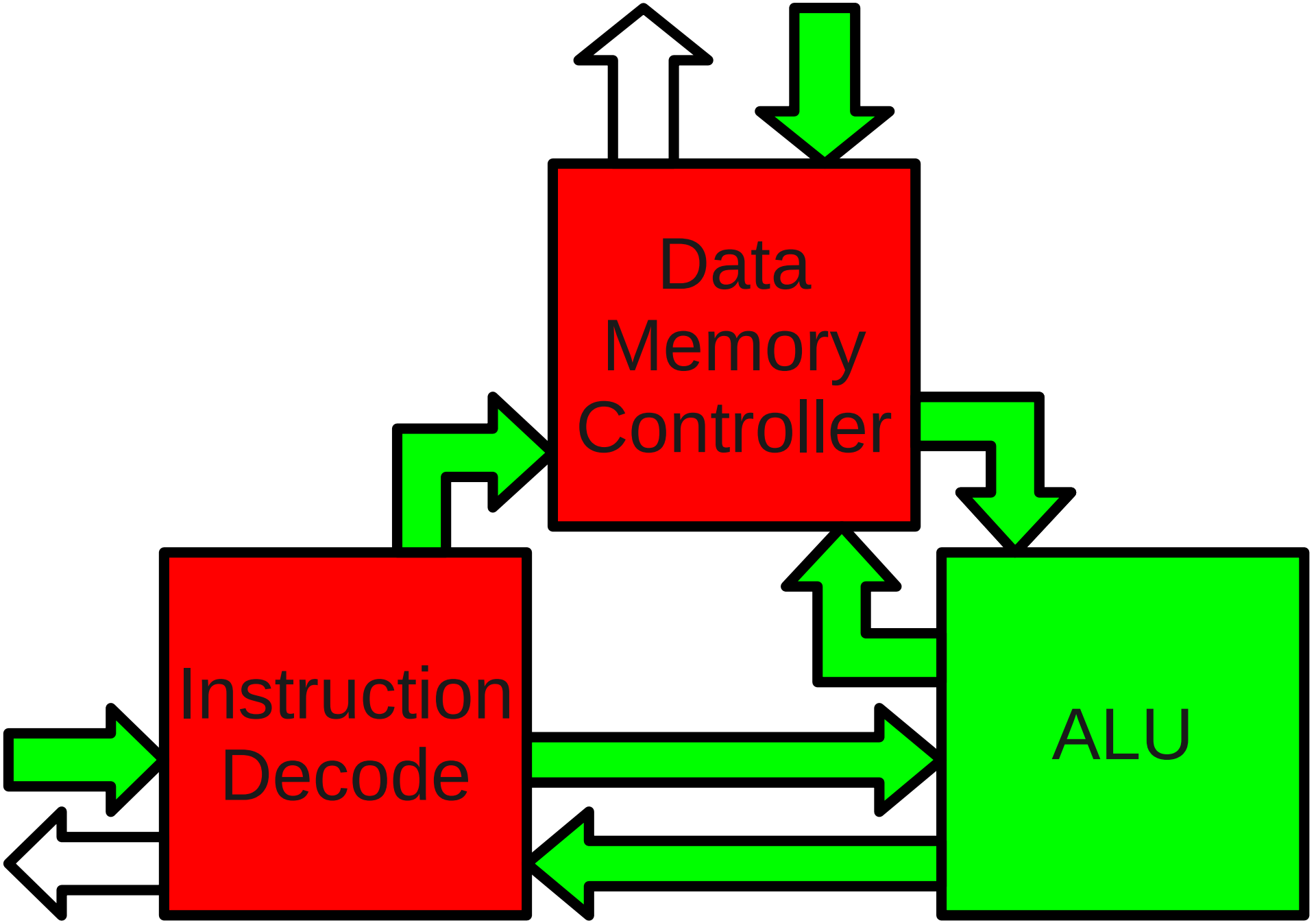

Data
Memory
Controller
Instruction
Decode
ALU

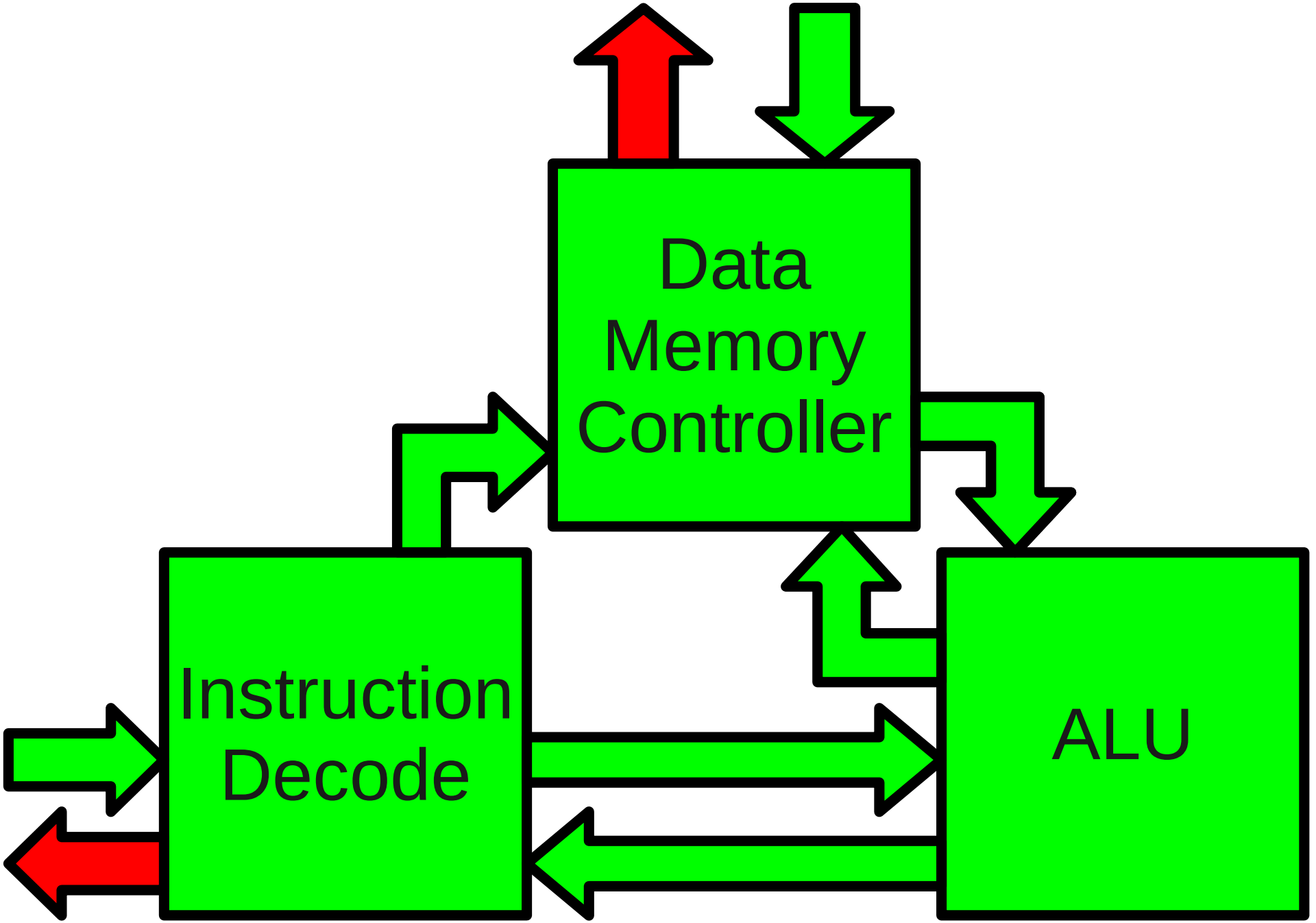

Data
Memory
Controller
Instruction
Decode
ALU


Data
Memory
Controller
Instruction
Decode
ALU


Data
Memory
Controller
Instruction
Decode
ALU

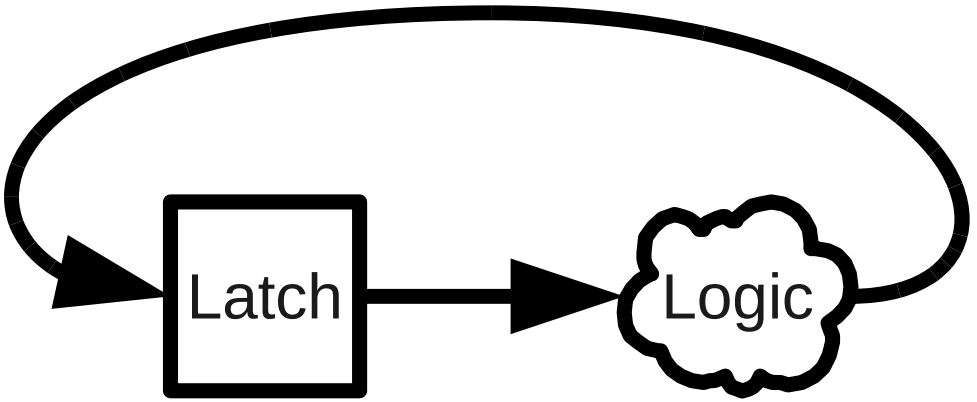

Data
Memory
Controller
Instruction
Decode
ALU

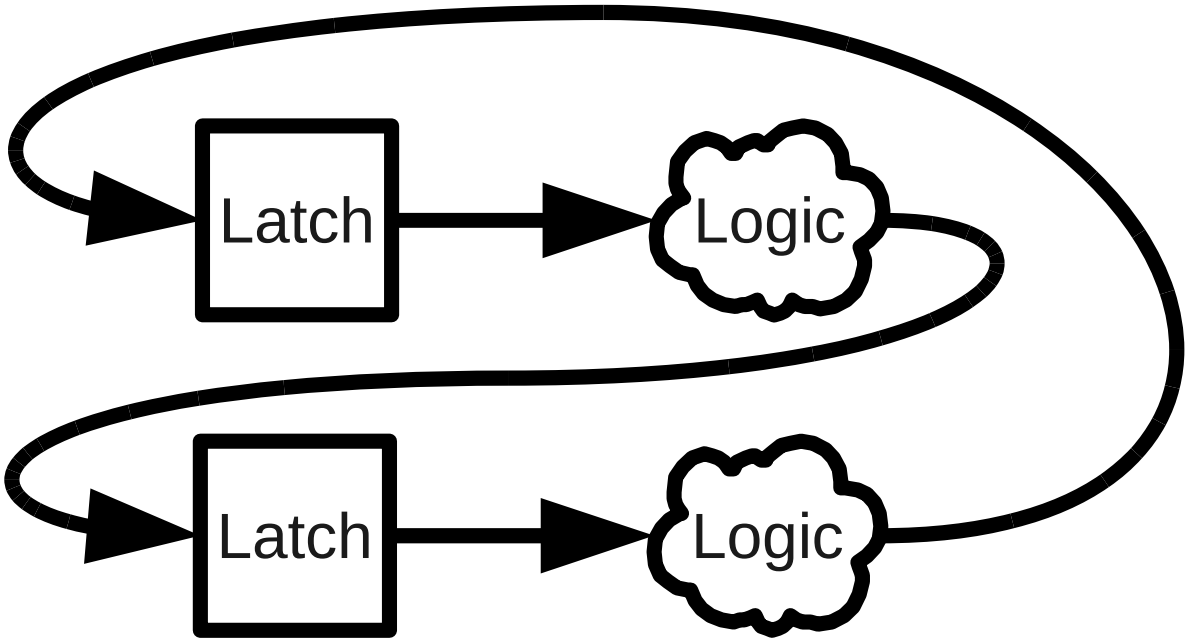

Data
Memory
Controller
Instruction
Decode
ALU

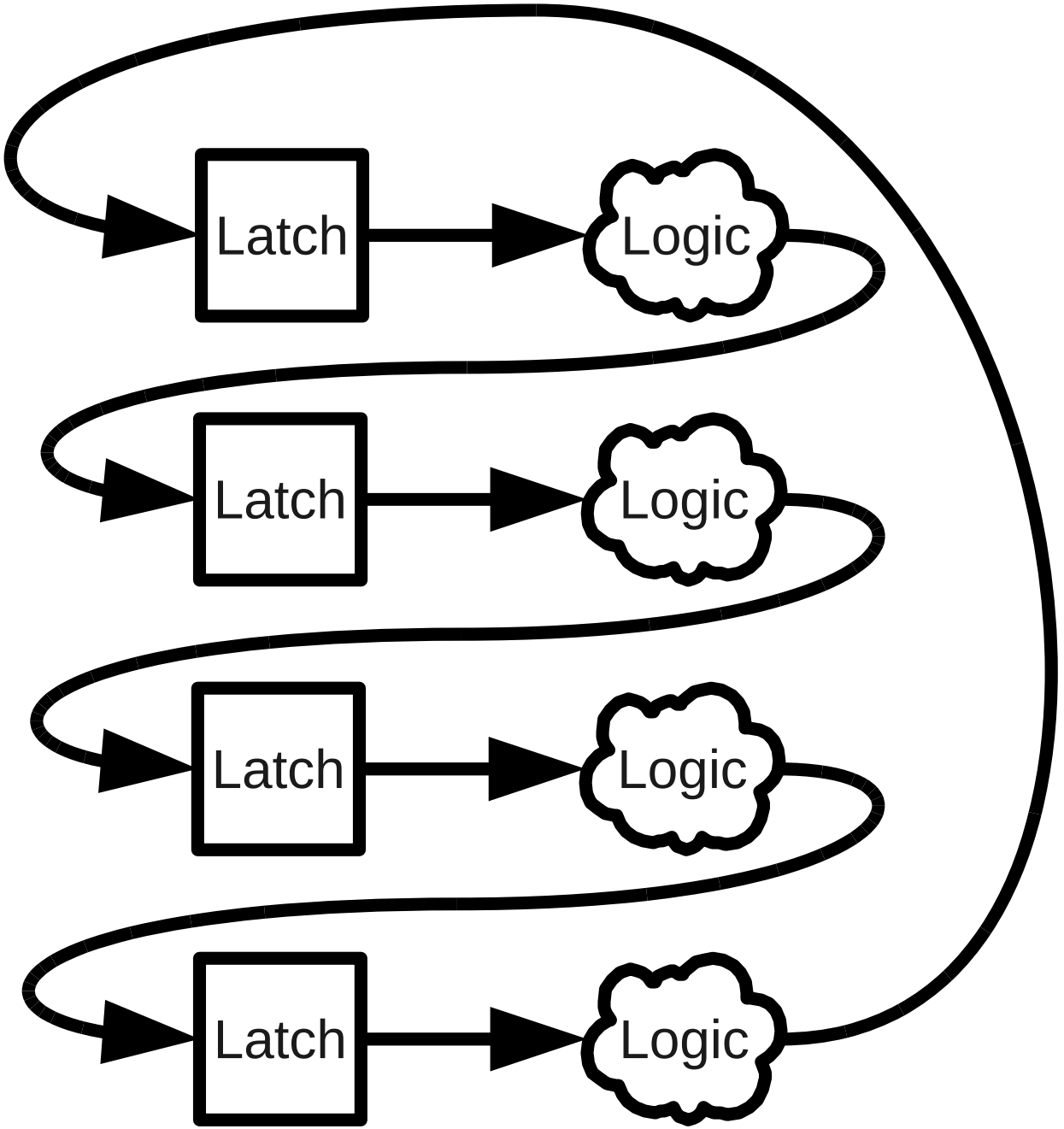

Data
Memory
Controller
Instruction
Decode
ALU

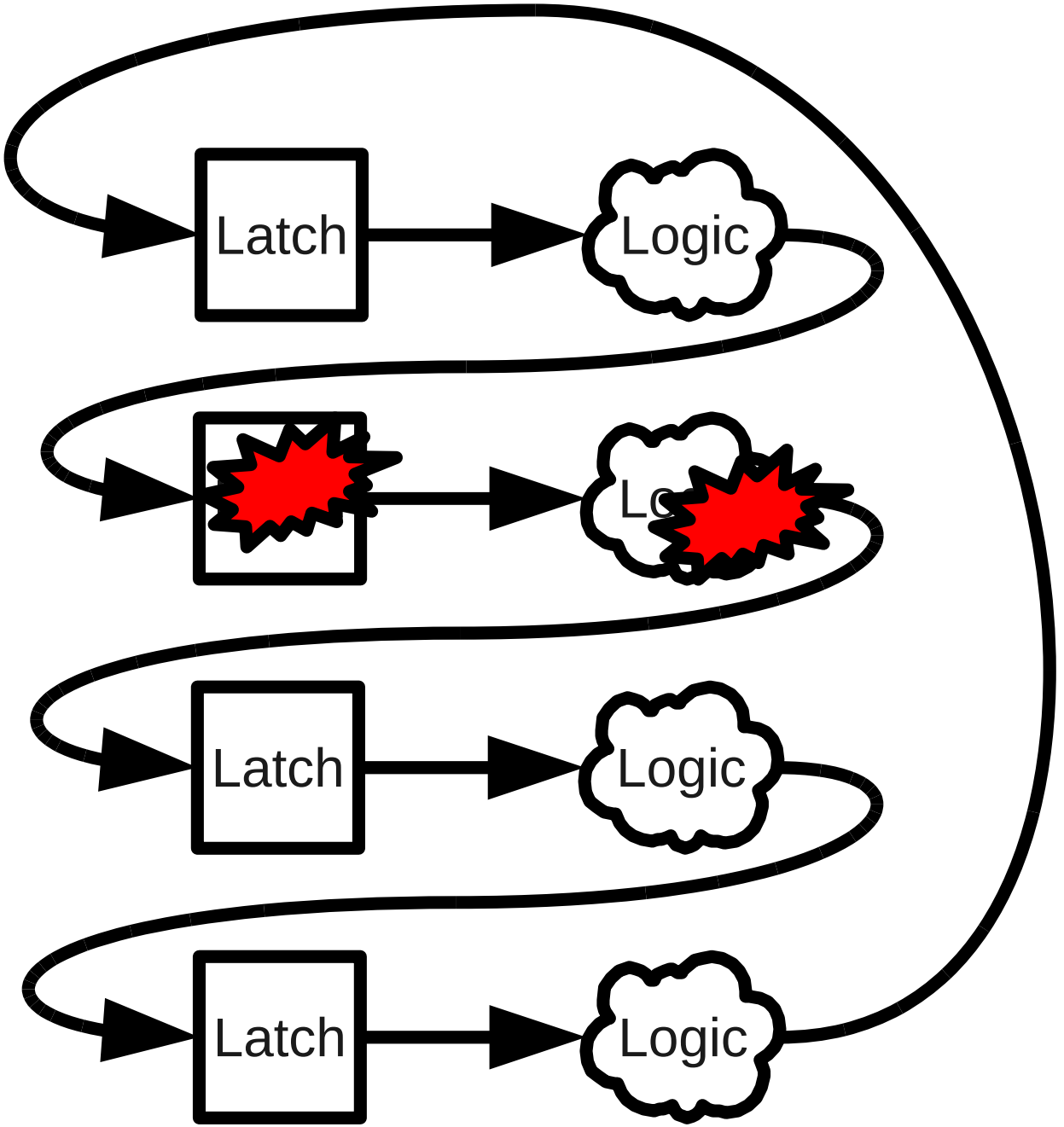
Performance wrapper

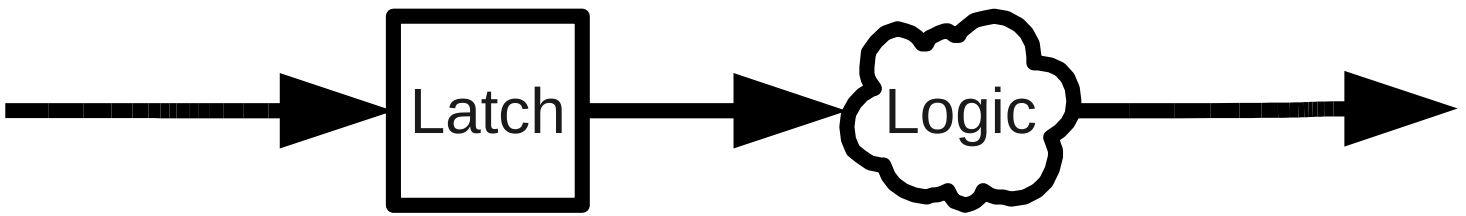
- Keep original structure and functional behaviour
- Design should remain functional in its synchronous form
- Wagging
  - With by-passable stages
- Add pipelining
- Add caching
  - Instruction cache
  - Data cache

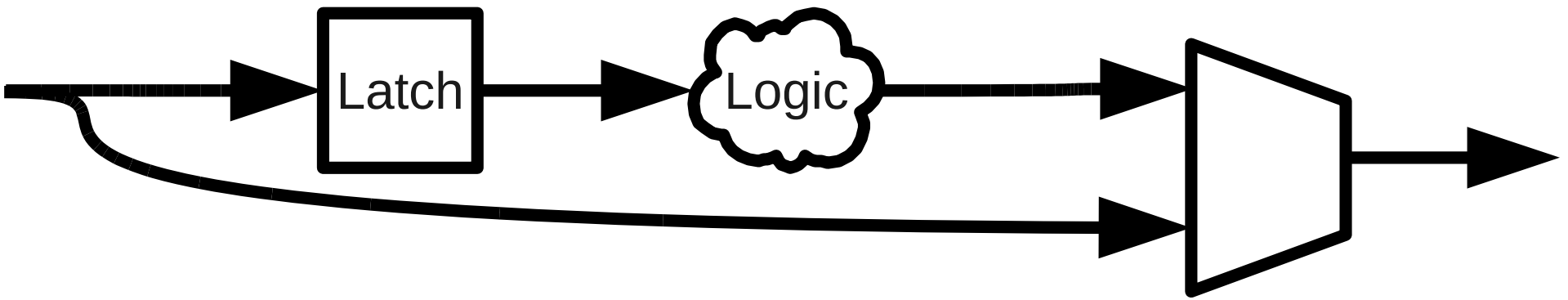

Latch
Logic

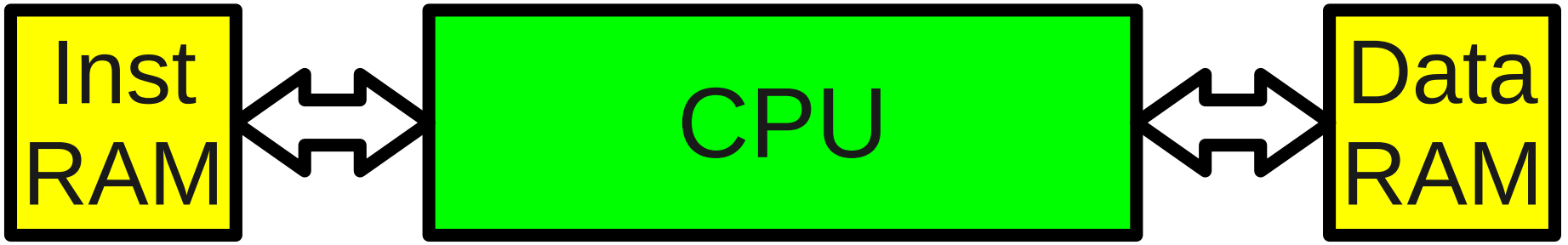

Latch
Logic
Latch
Logic

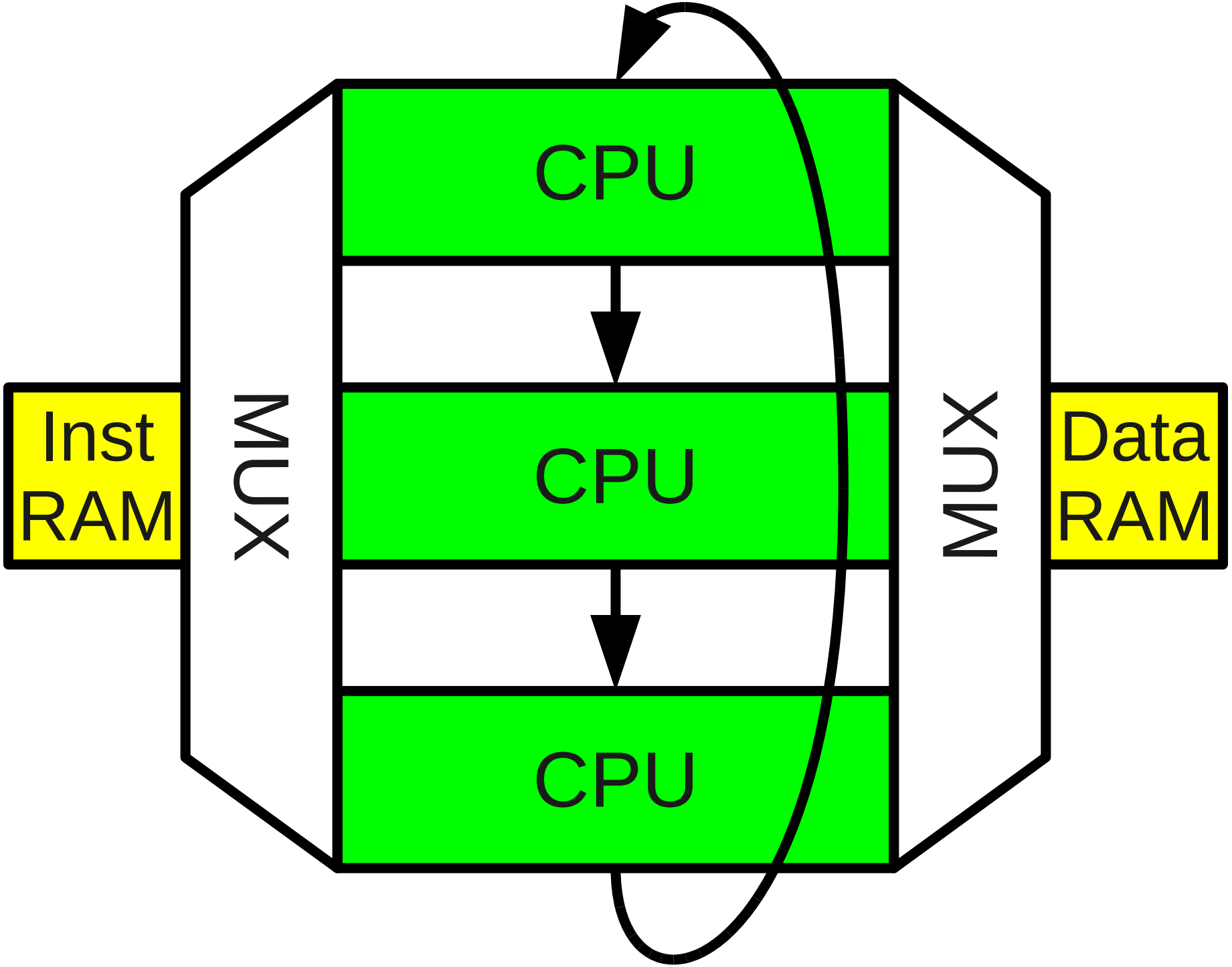

Latch
Logic
Latch
Logic
Latch
Logic
Latch
Logic

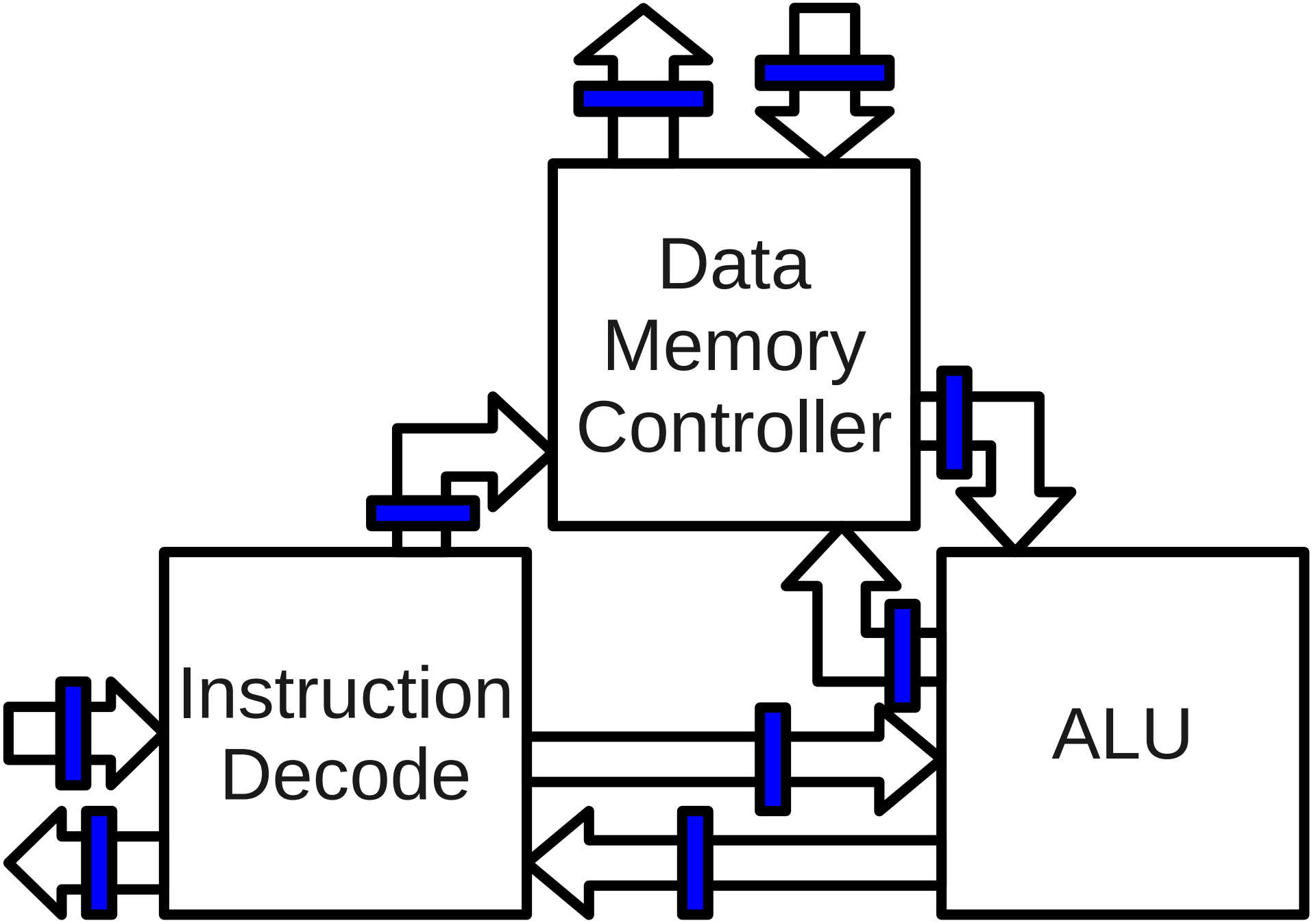

Latch
Logic
Latch
Logic
Latch
Logic

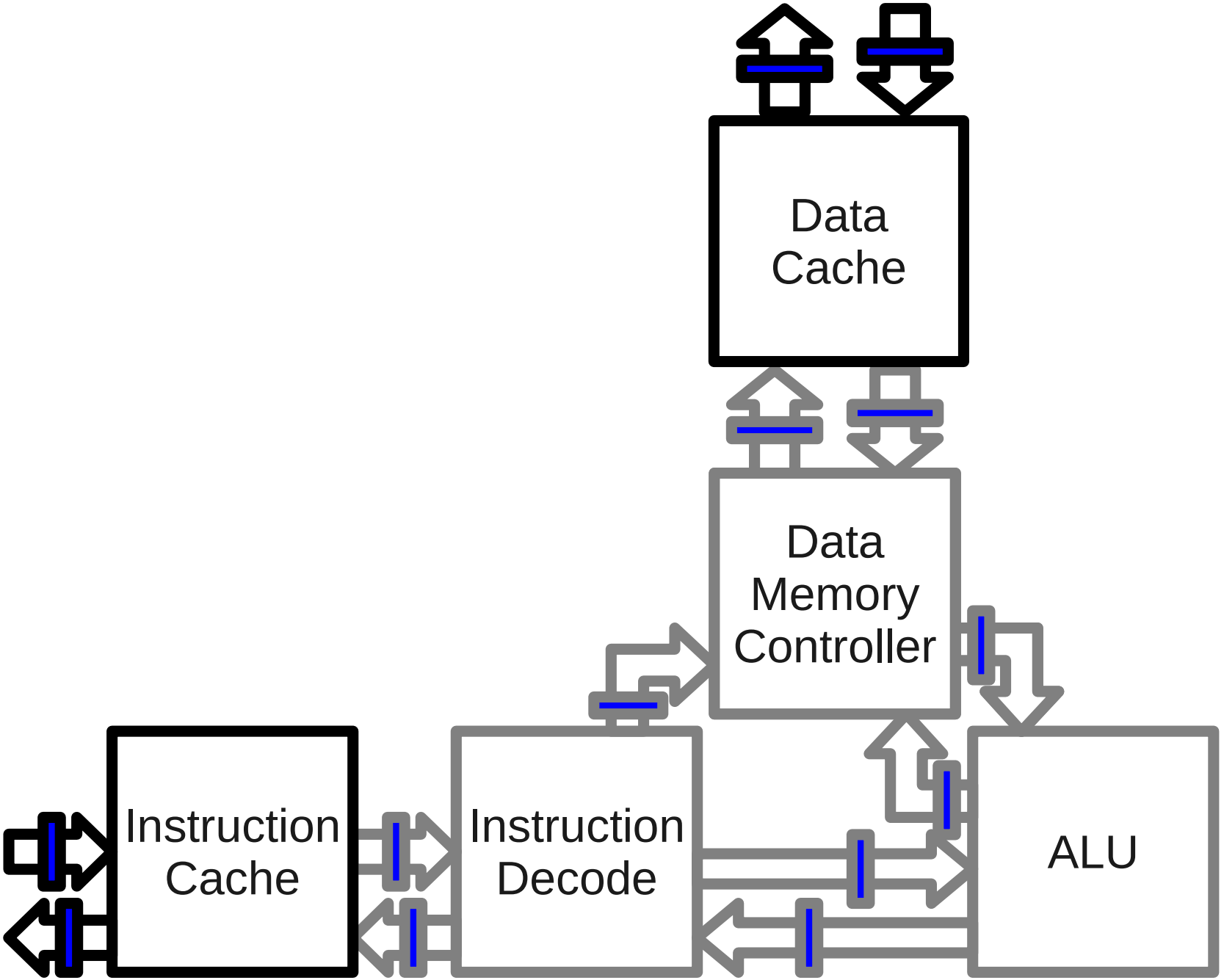

Latch
Logic


Latch
Logic

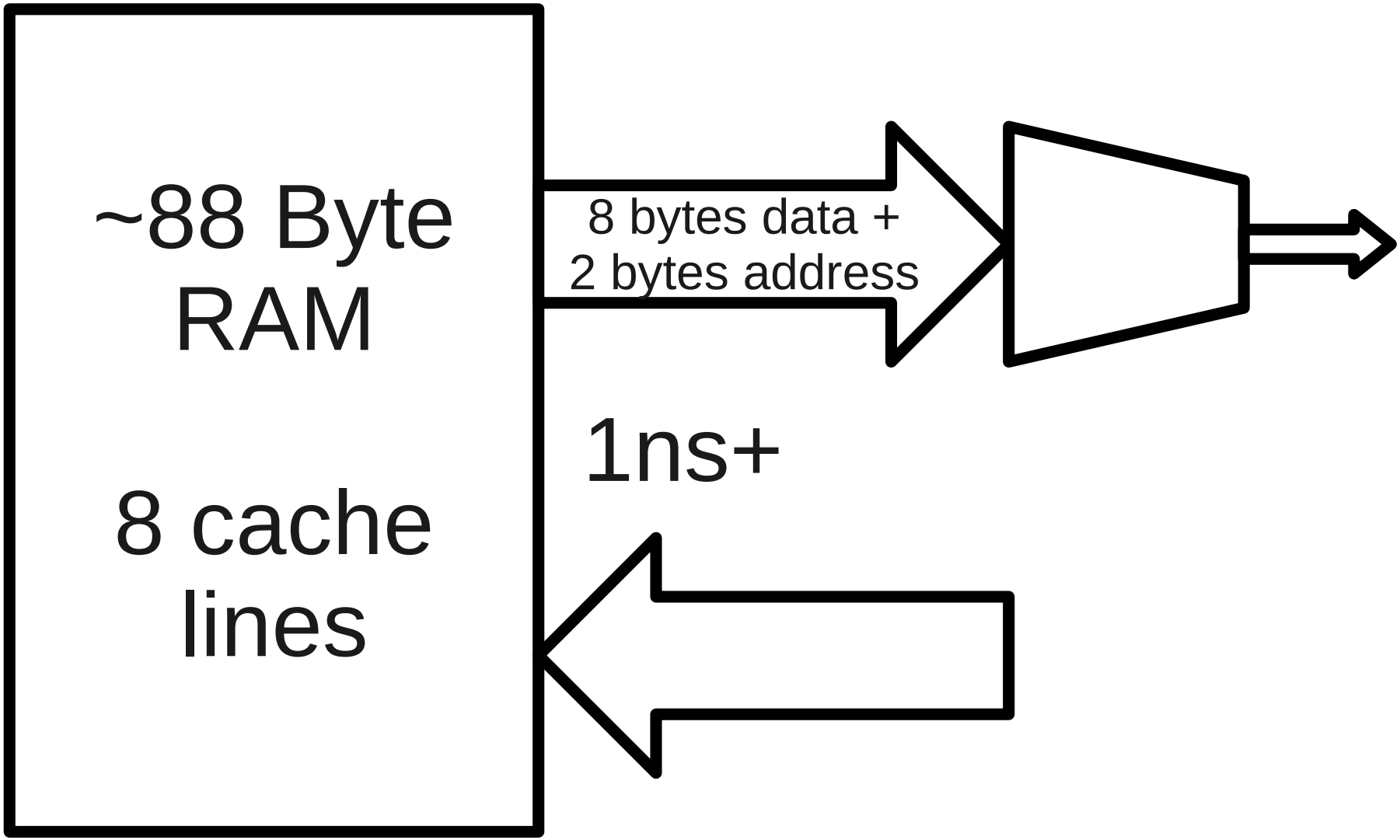

Inst
RAM
CPU
Data
RAM


CPU
CPU
CPU
Inst
RAM
MUX
MUX
Data
RAM

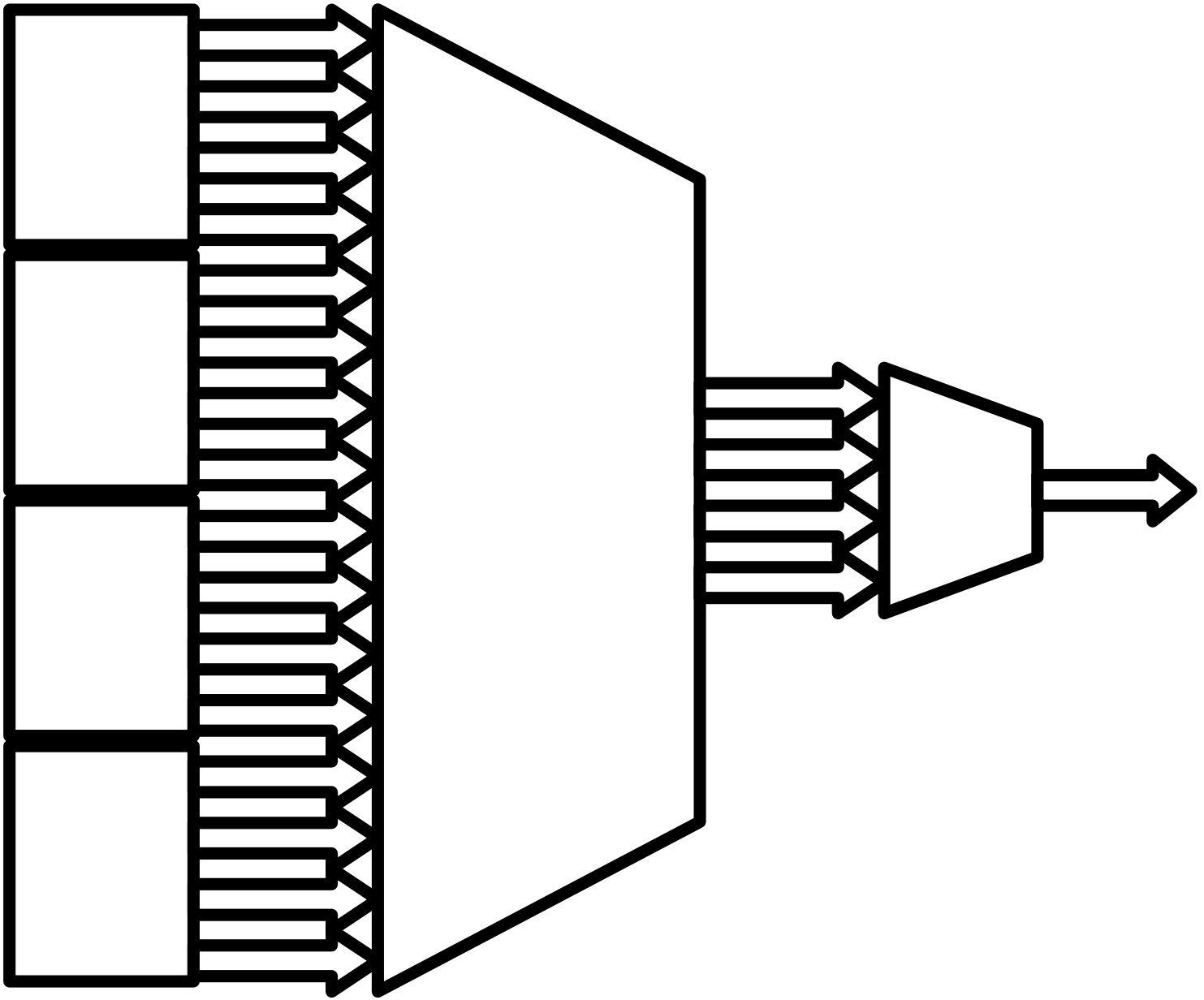

Data
Memory
Controller
Instruction
Decode
ALU


Data
Cache
Data
Memory
Controller
Instruction
Cache
Instruction
Decode
ALU

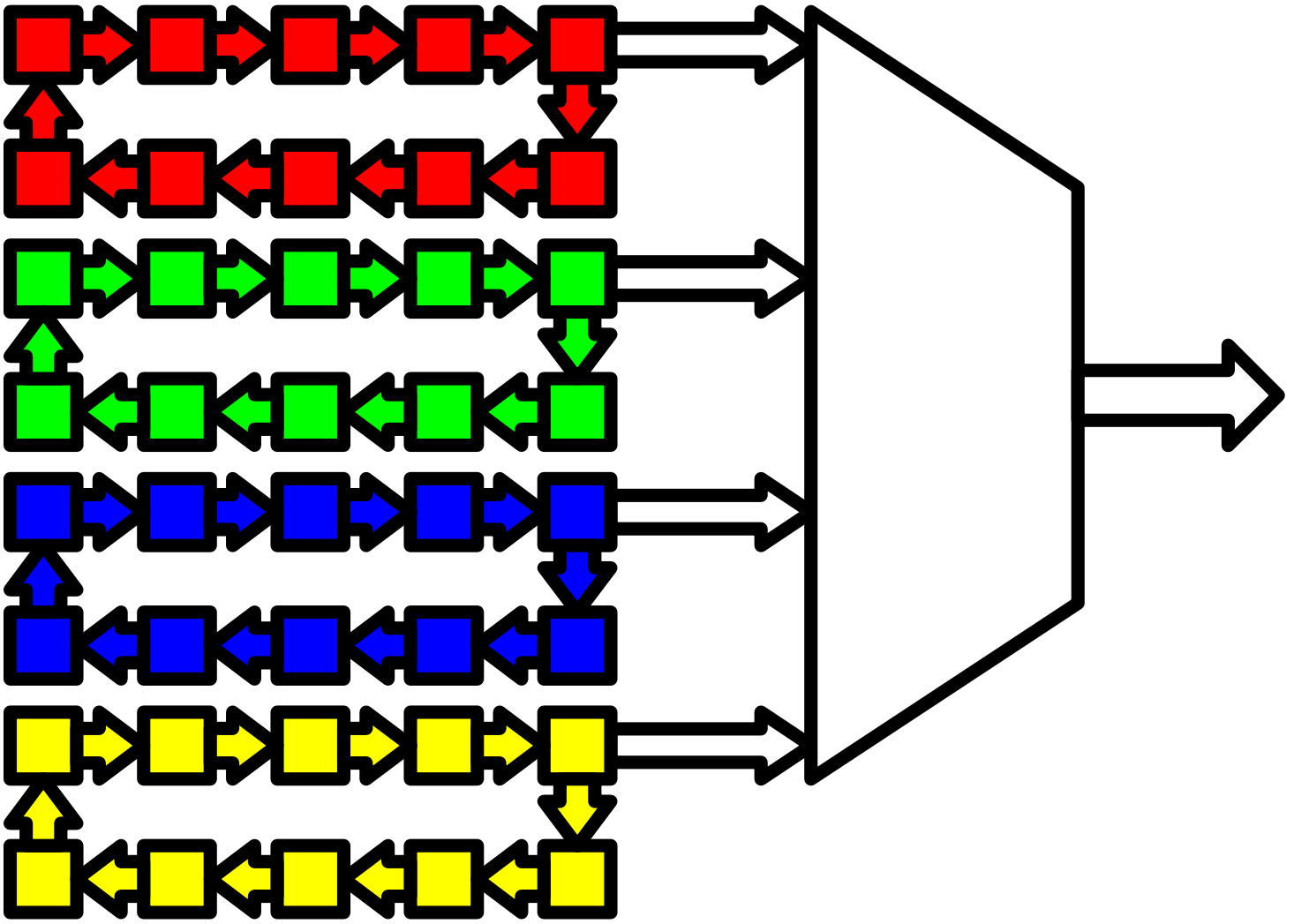
Instruction cache option 1



Instruction cache option 2



Instruction cache option 3

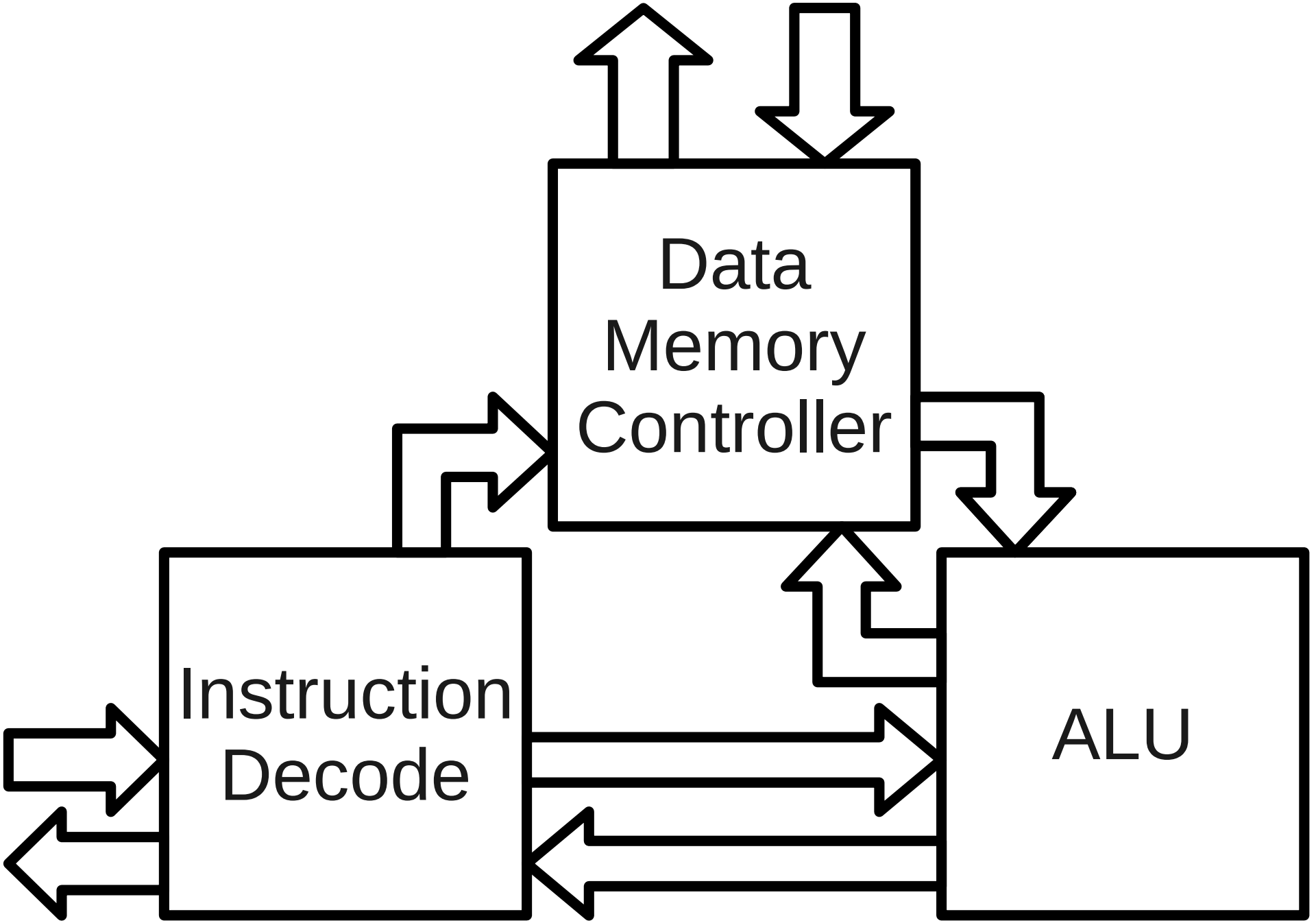


Data cache

- Register bank cache
  - 8 registers
  - Stack Pointer
- 2 entry stack cache
  - Snoops instruction stream
    - Detects push and call operations
      - Flushes one entry
  - Allows all instructions to execute in one cycle per instruction stream byte read

Comparisons (How to cheat)

- Handshake solutions
  - State performance in MHz equivalent
    - 60.5 MHz – Woot!
      - 6 MIPS
  - ~100 times faster
- Caltech
  - Only measure the performance of 2 instructions
    - Accumulate and NOP
    - Don't memory map the accumulator
    - ~100 MIPS


Data
Memory
Controller
Instruction
Decode
ALU

Tape-out

- Implemented in 130nm tech
- 1st tape-out was “cancelled”
- 2nd tape-out 22nd of November
-