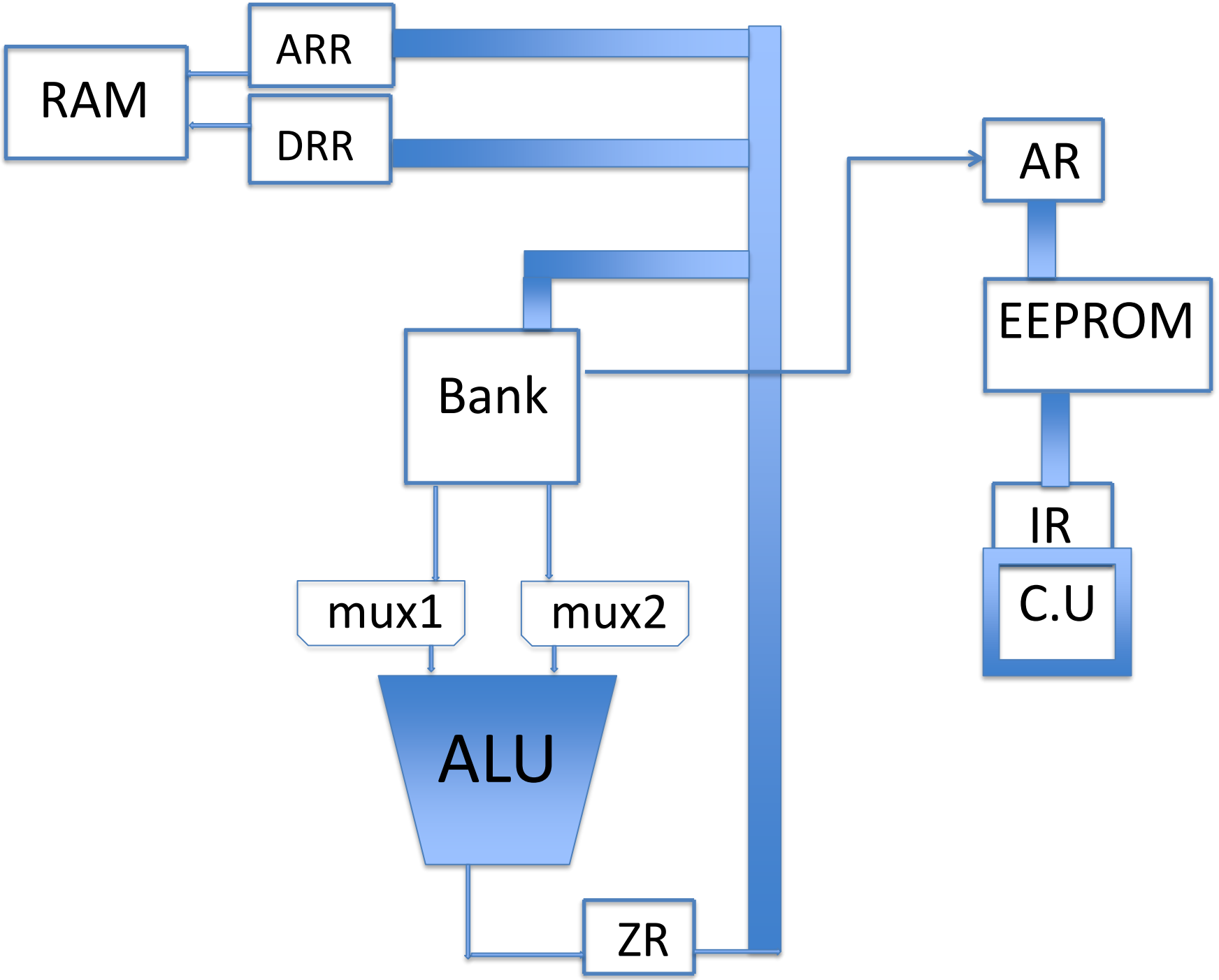
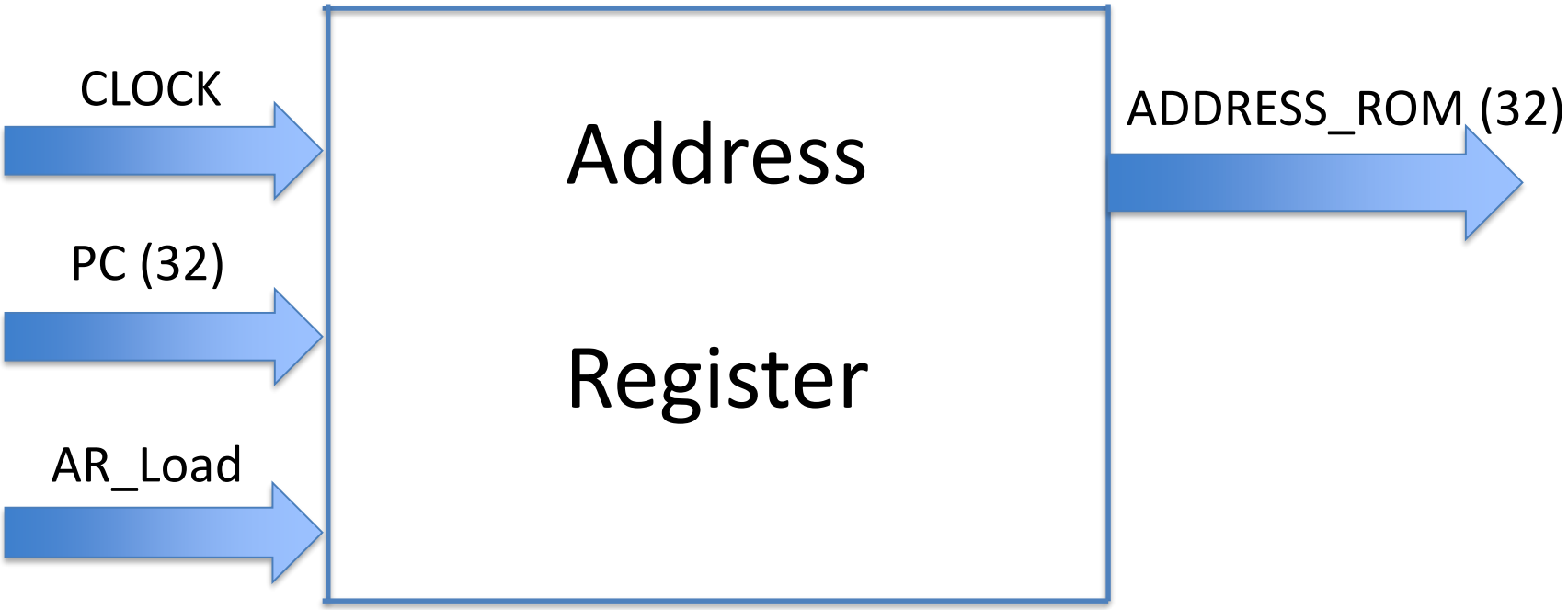
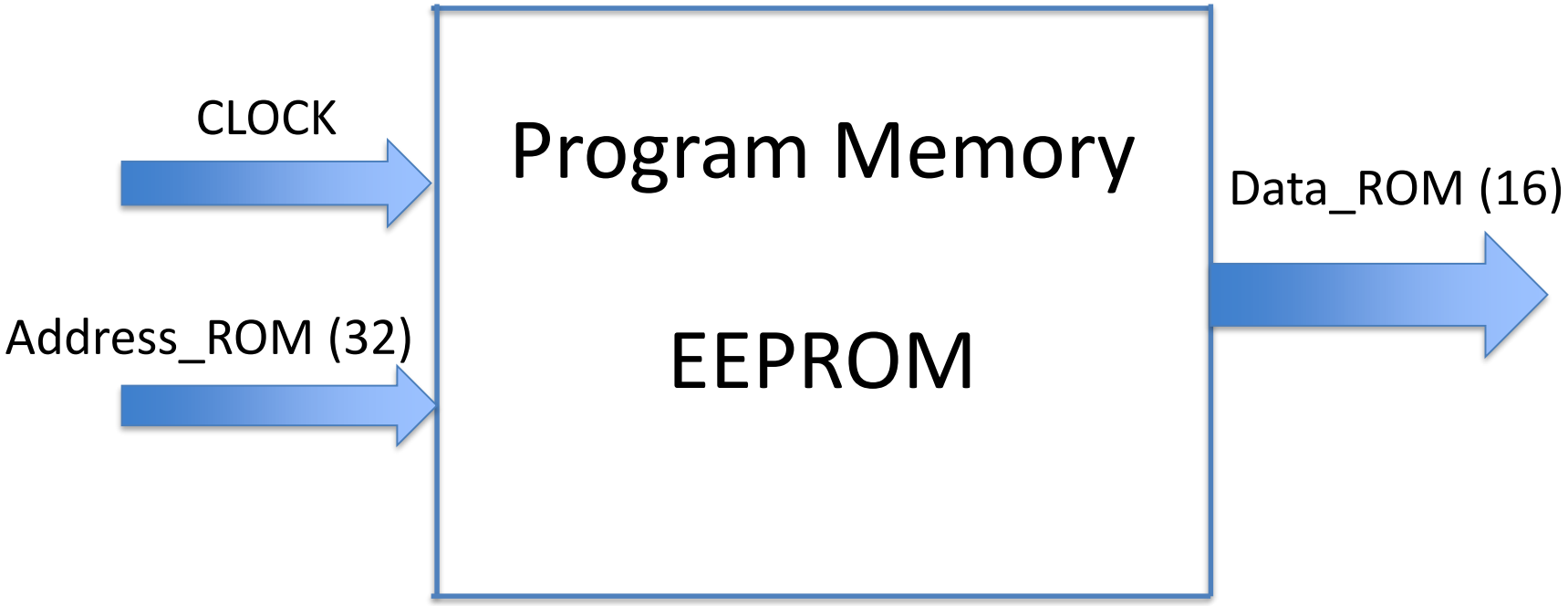
ARM

Cortex-M0

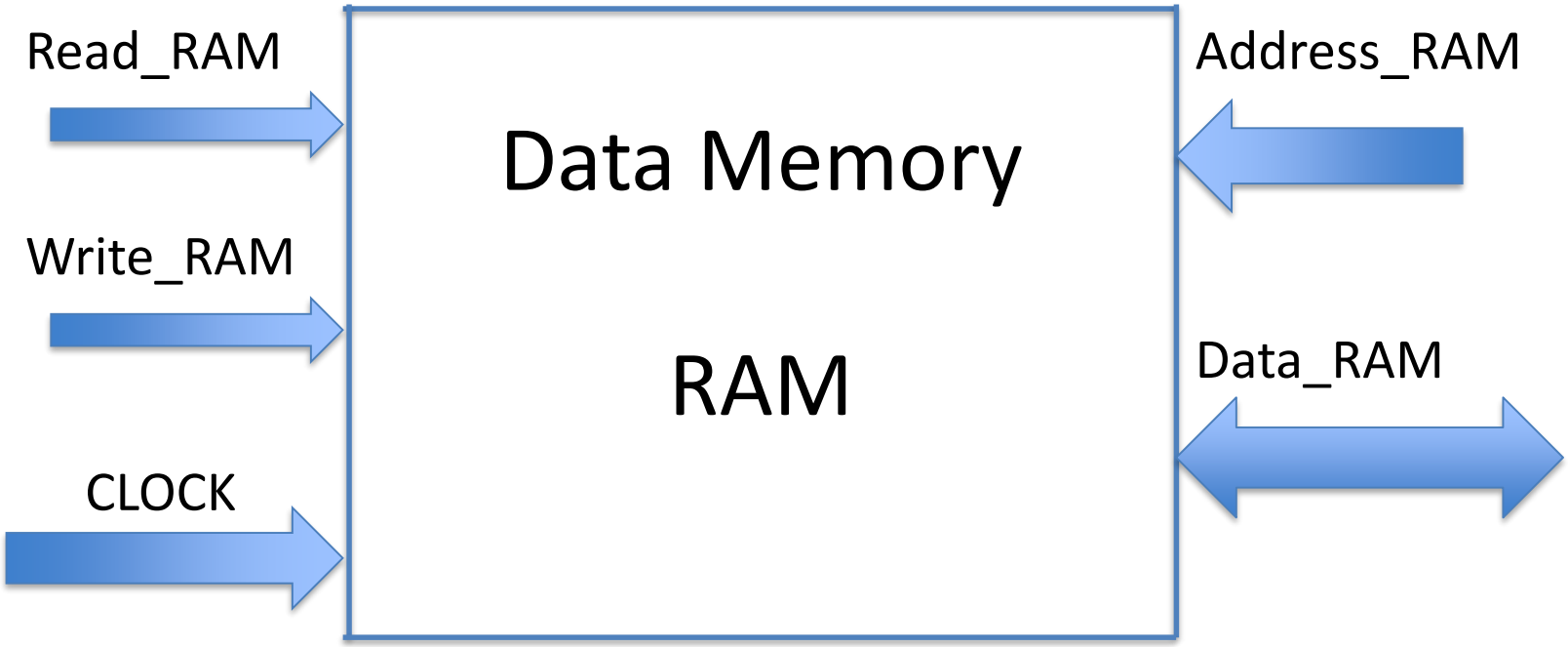
Processor

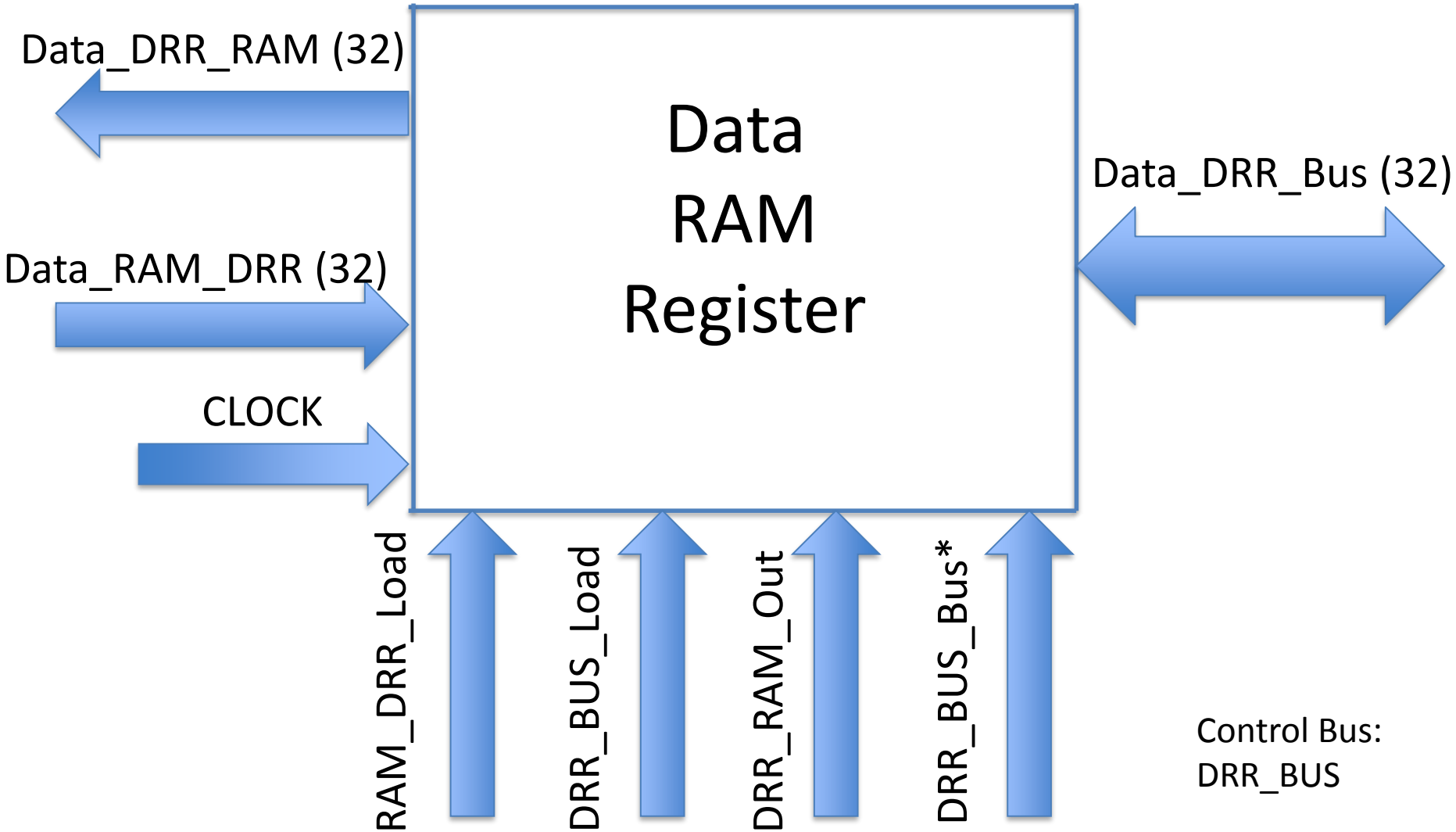

RAM
ARR
DRR
AR
EEPROM
Bank
IR
C.U
mux1
mux2
ALU
ZR


CLOCK
PC (32)
AR_Load
Address
Register
ADDRESS_ROM (32)

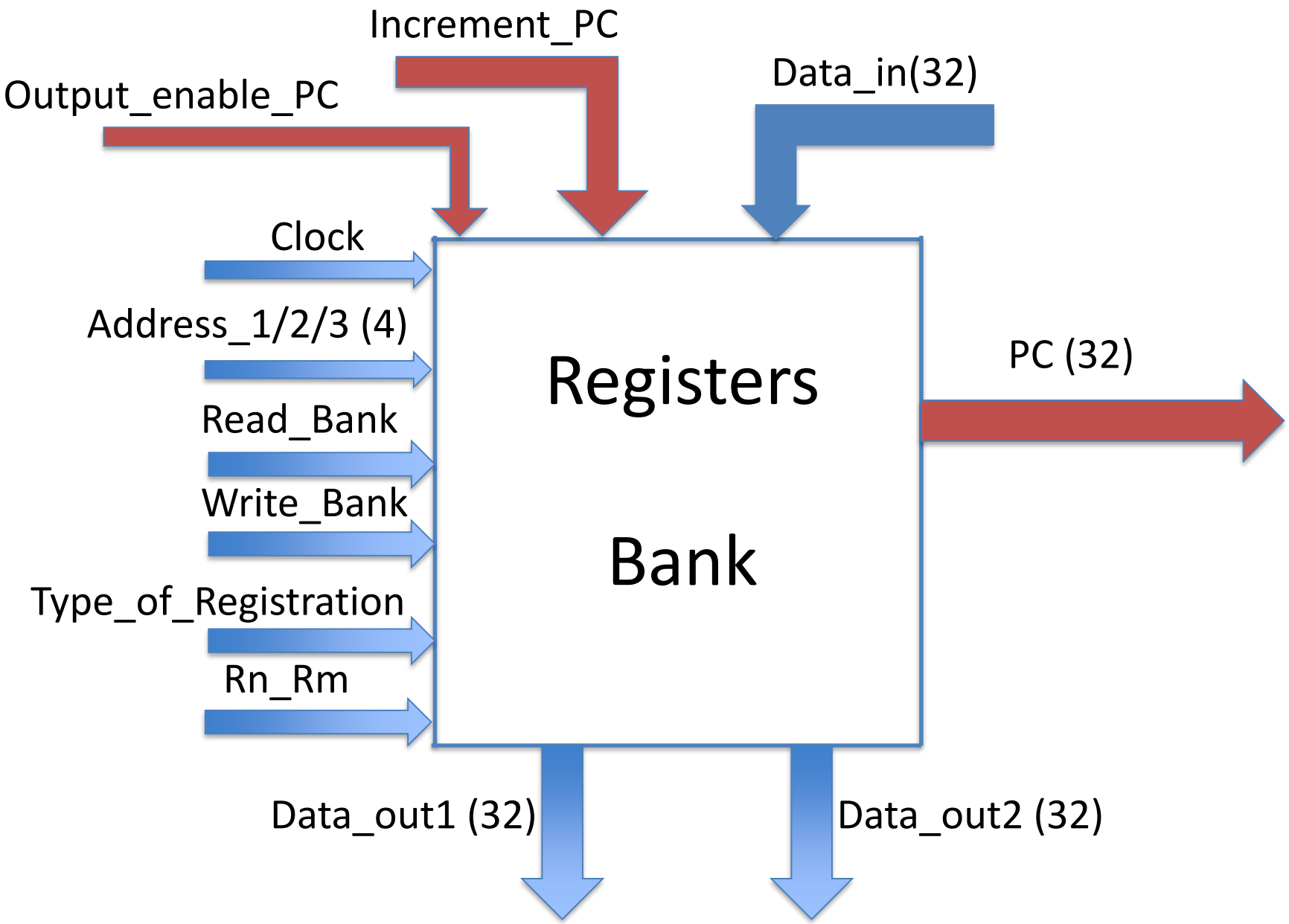


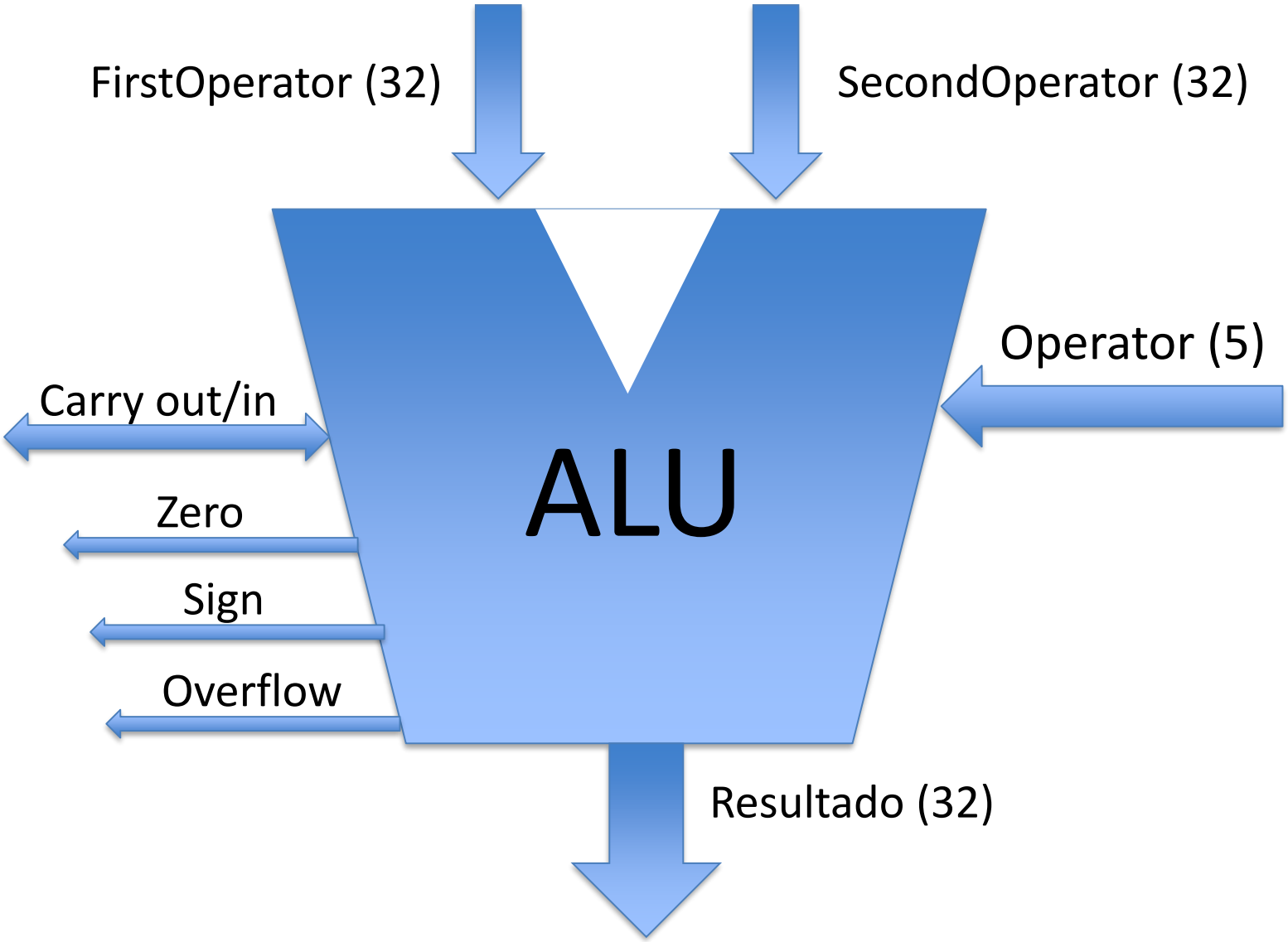
* Dejar Read Enable siempre a Vcc
* Write Enable Gnd
* Output Enable siempre Vcc


Read_RAM
Write_RAM
CLOCK
Data Memory
RAM
Address_RAM
Data_RAM

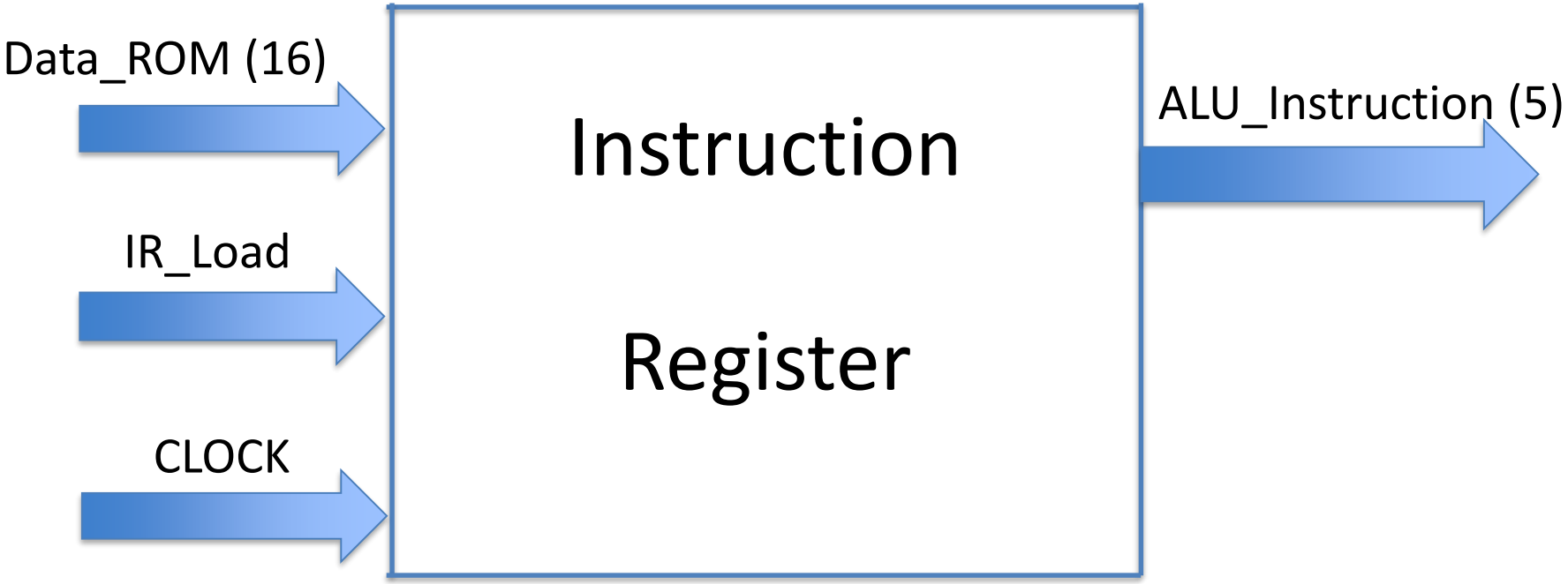

Data_DRR_RAM (32)
Data_RAM_DRR (32)
CLOCK
Data
RAM
Register
Data_DRR_Bus (32)
RAM_DRR_Load
DRR_BUS_Load
DRR_RAM_Out
DRR_BUS_Bus*
Control Bus:
DRR_BUS

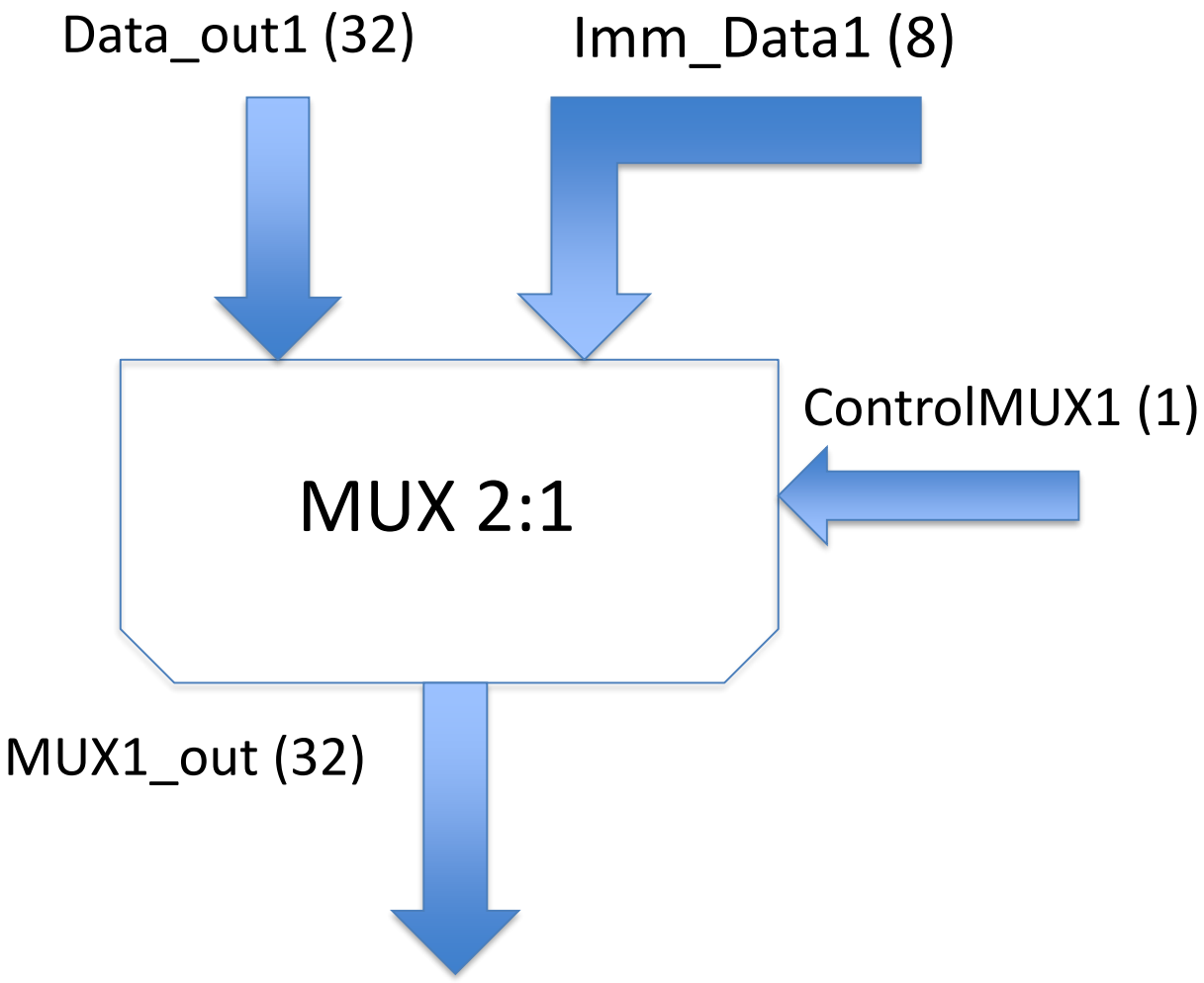

CLOCK
Address_out (32)
Address
RAM
Register
Address_in (32)
ARR_Load

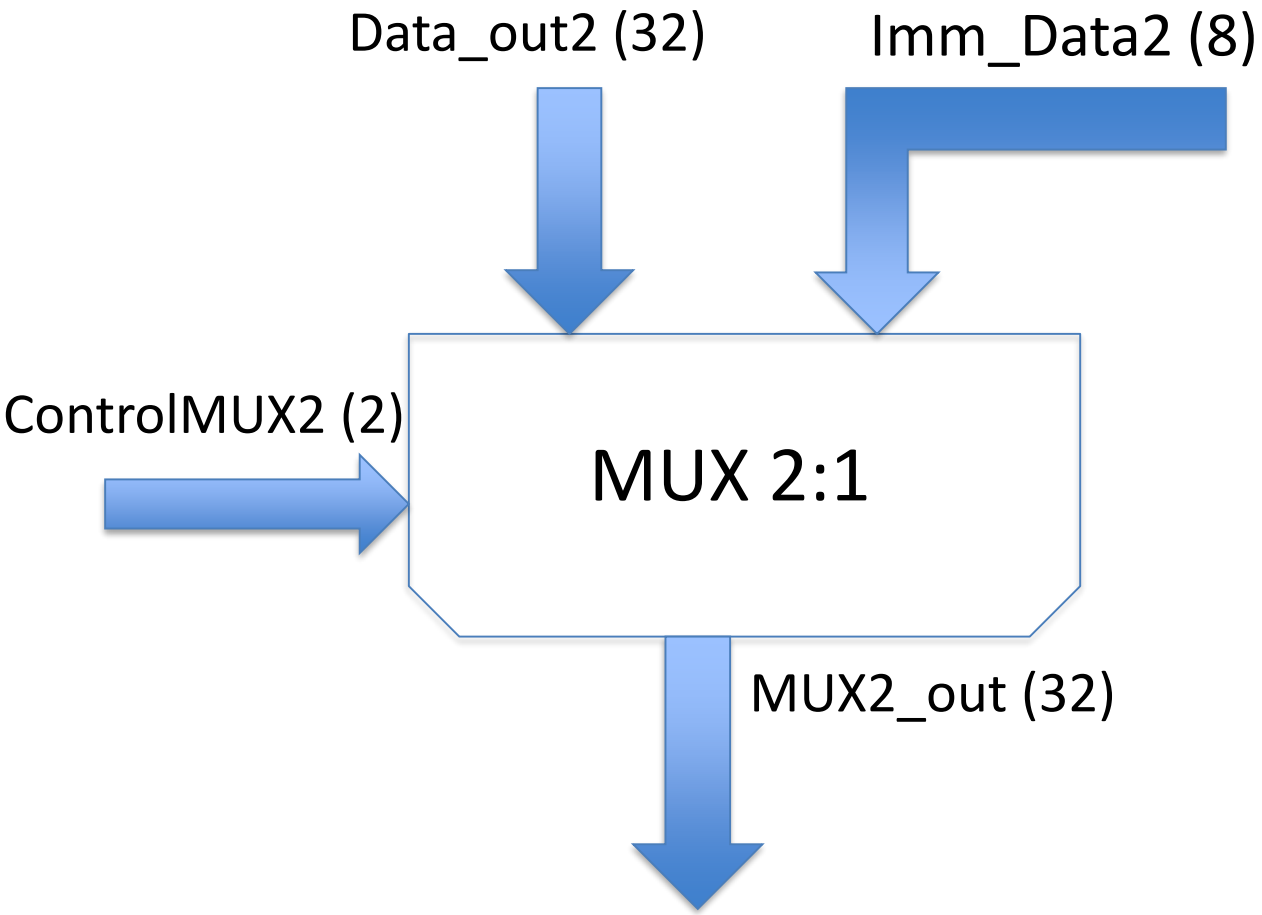

Increment_PC
Output_enable_PC
Data_in(32)
Clock
Address_1/2/3 (4)
Registers
PC (32)
Read_Bank
Write_Bank
Bank
Type_of_Registration
Rn_Rm
Data_out1 (32)
Data_out2 (32)

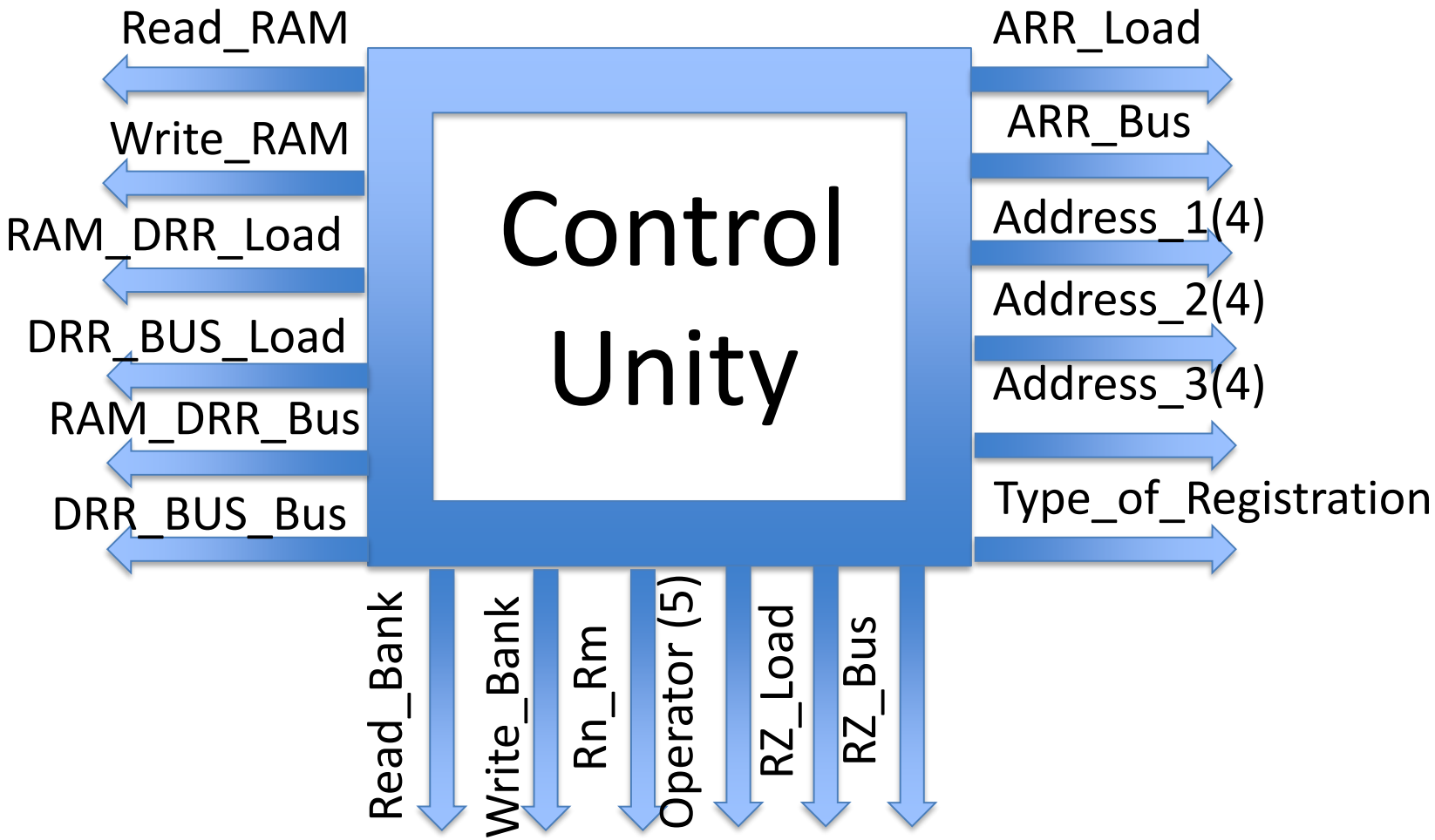

FirstOperator (32)
SecondOperator (32)
Operator (5)
Carry out/in
ALU
Zero
Sign
Overflow
Resultado (32)

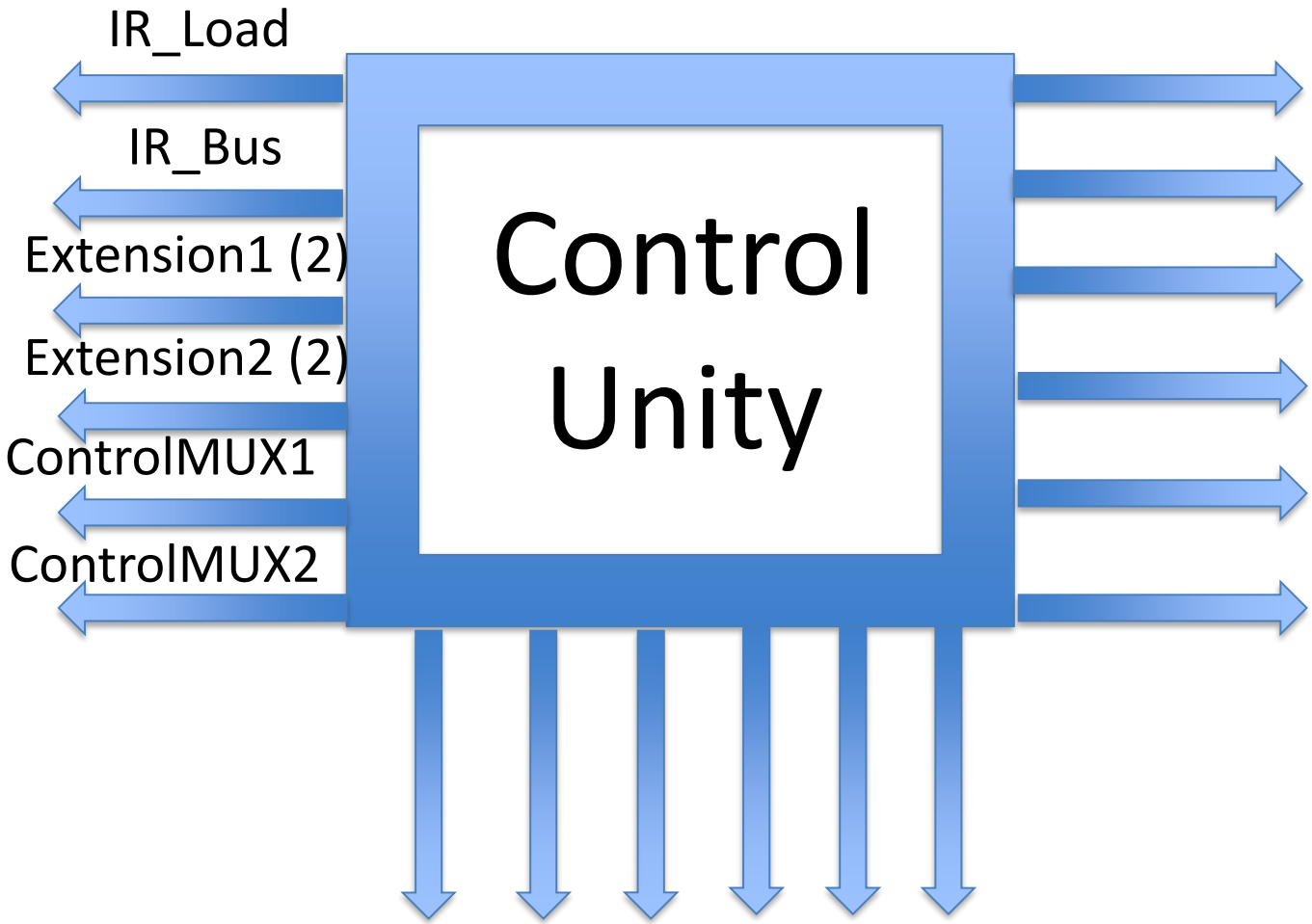
Control Bus:
RZ_BUS

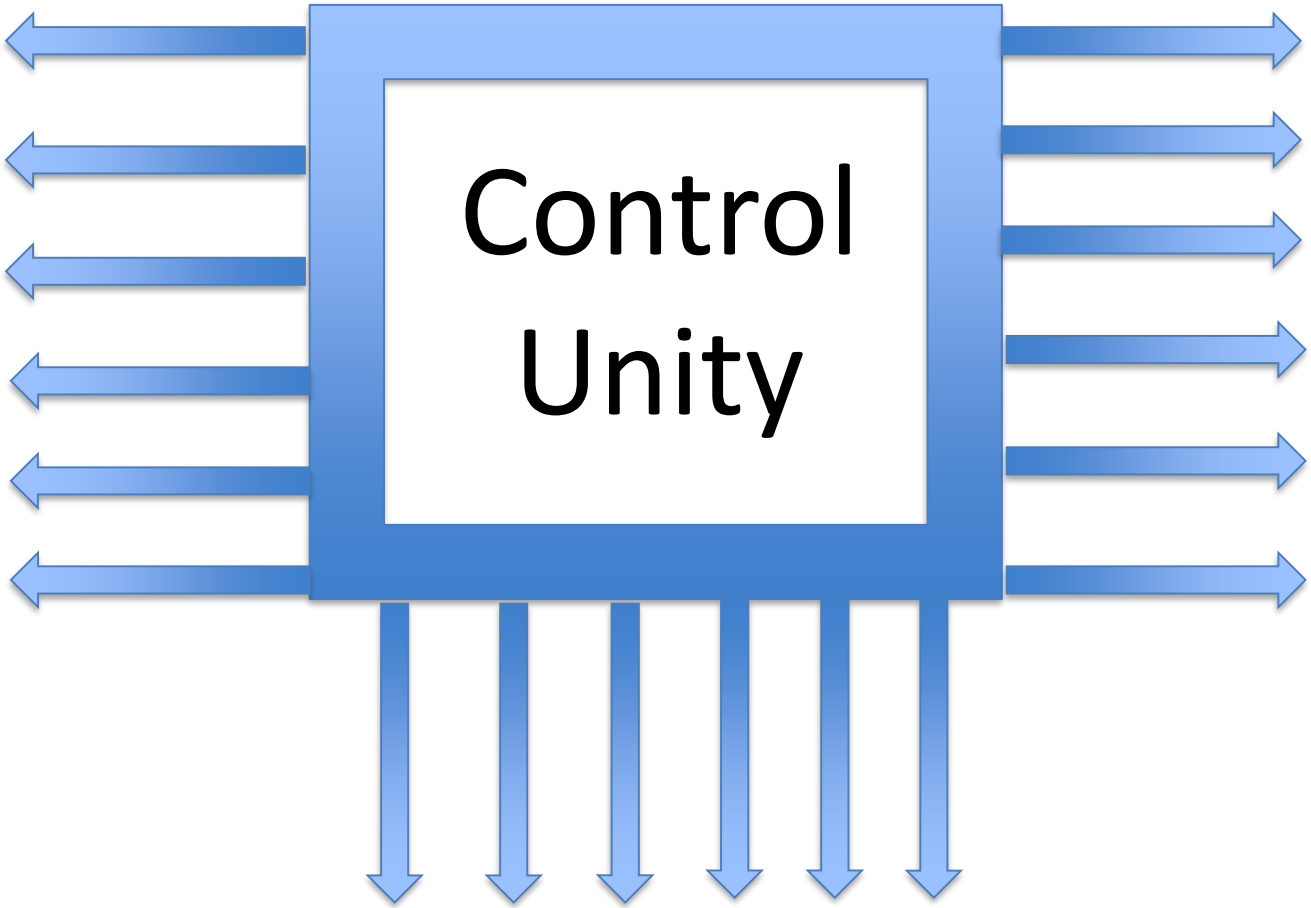

Data_ROM (16)
IR_Load
CLOCK
Instruction
Register
ALU_Instruction (5)

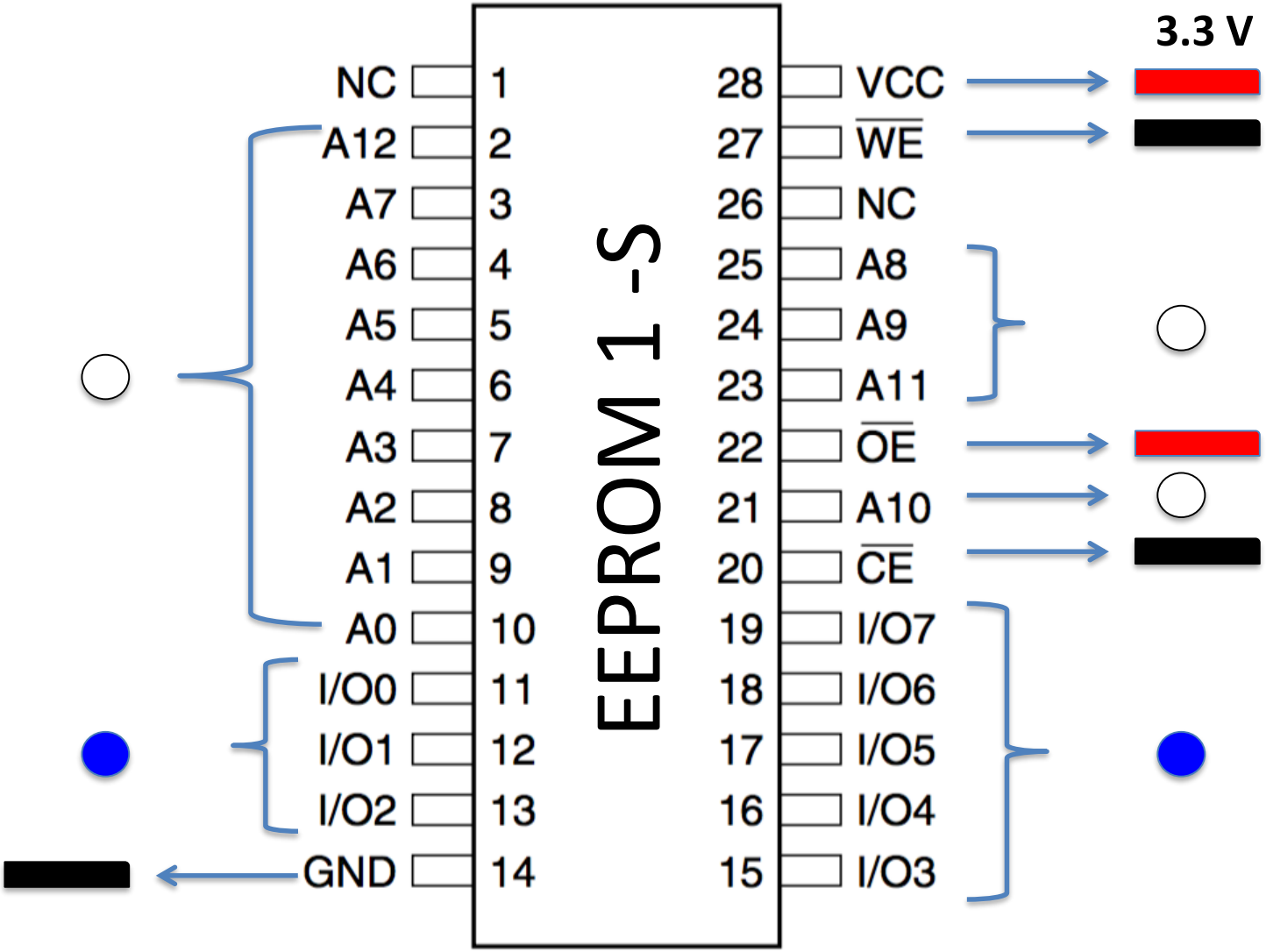

Data_out1 (32)
Imm_Data1 (8)
ControlMUX1 (1)
MUX 2:1
MUX1_out (32)


Data_out2 (32)
Imm_Data2 (8)
ControlMUX2 (2)
MUX 2:1
MUX2_out (32)

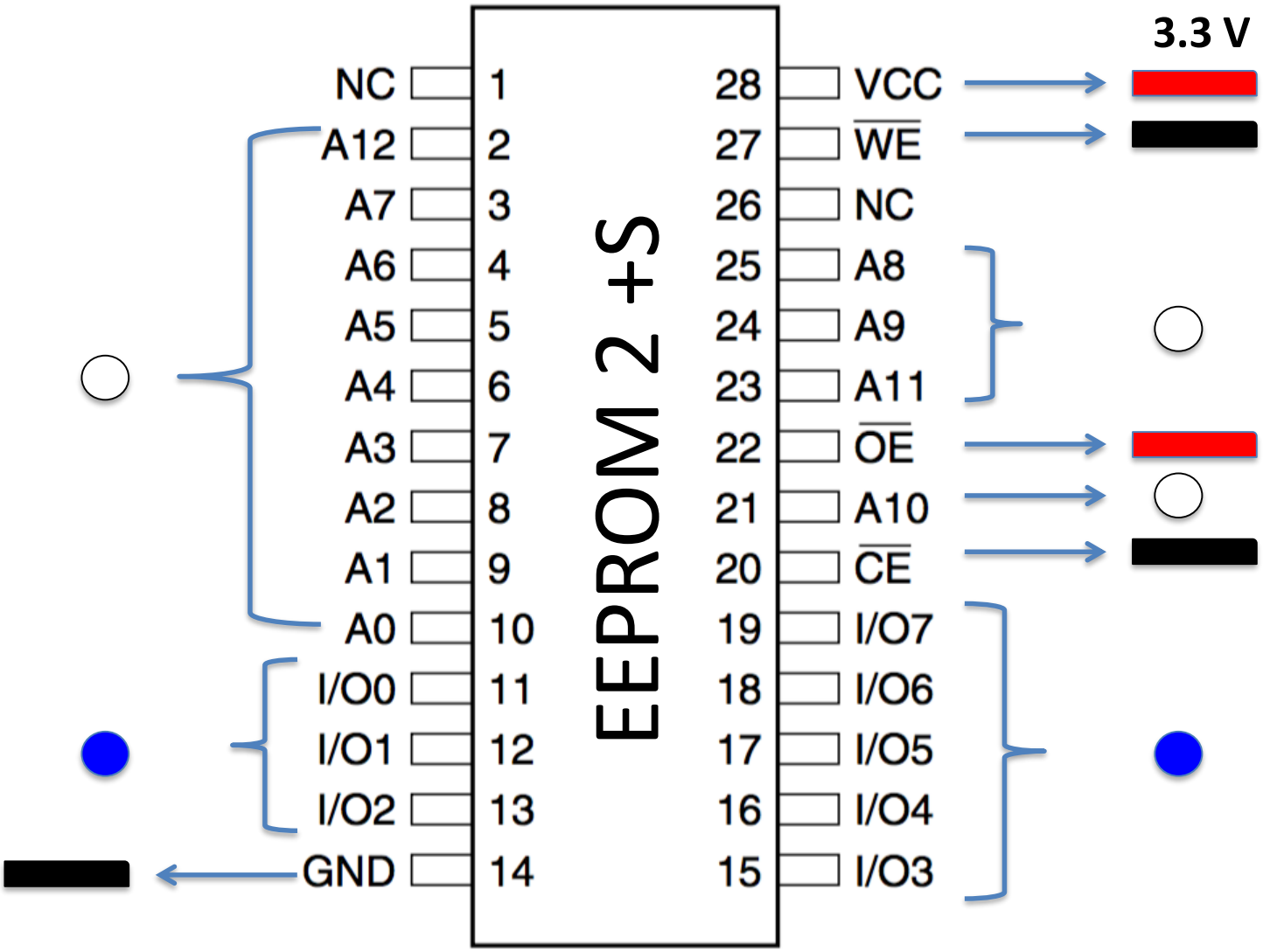

Read_RAM
Write_RAM
RAM_DRR_Load
DRR_BUS_Load
RAM_DRR_Bus
DRR_BUS_Bus
Control Unity
ARR_Load
ARR_Bus
Address_1(4)
Address_2(4)
Address_3(4)
Type_of_Registration
Read_Bank
Write_Bank
Rn_Rm
Operator (5)
RZ_Load
RZ_Bus


IR_Load
IR_Bus
Extension1 (2)
Extension2 (2)
ControlMUX1
ControlMUX2
Control
Unity

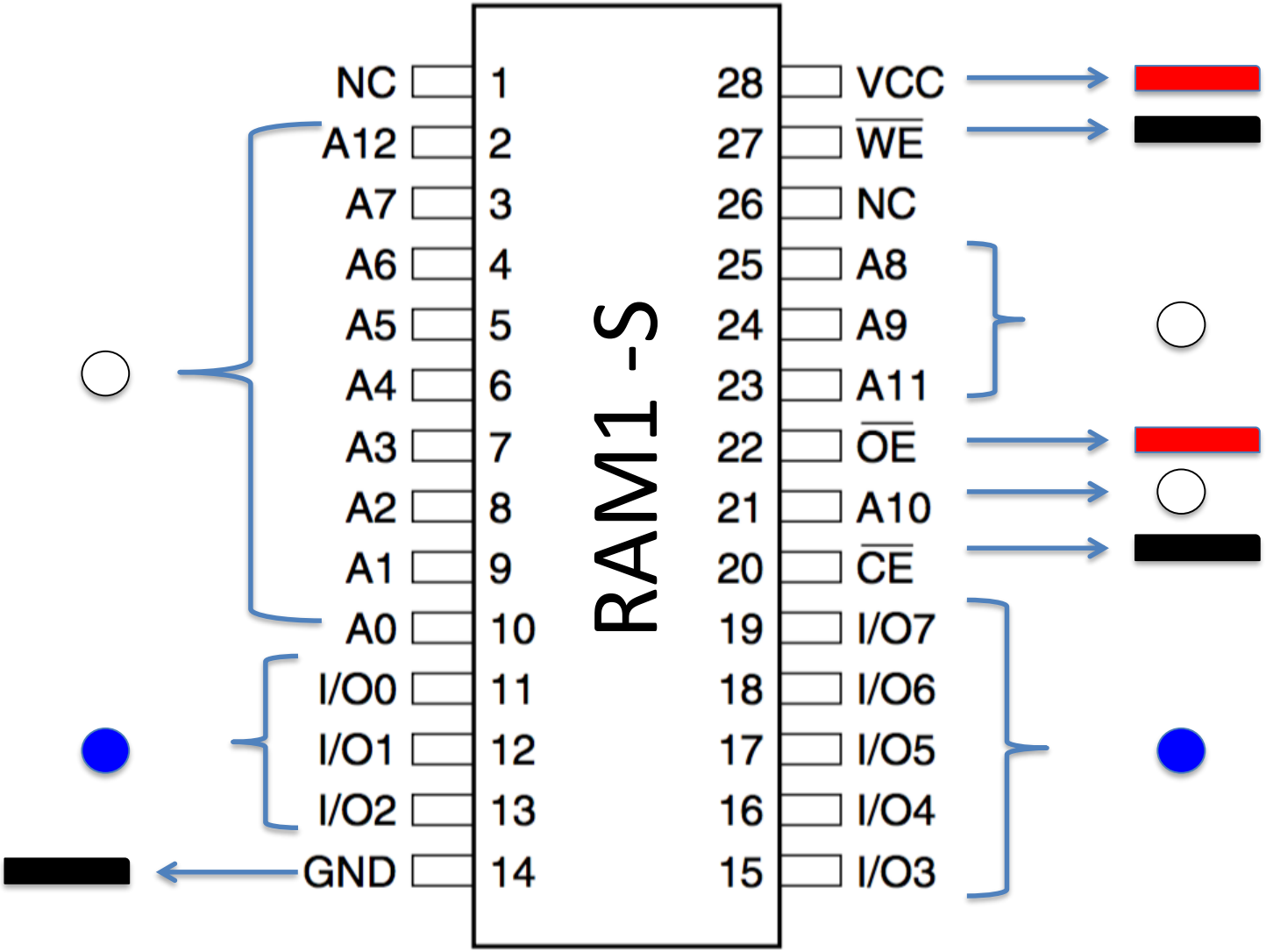

Control
Unity


3.3 V
EEPROM 1 -S
NC 1
A12 2
A7 3
A6 4
A5 5
A4 6
A3 7
A2 8
A1 9
A0 10
I/O0 11
I/O1 12
I/O2 13
GND 14
28 VCC
27 WE
26 NC
25 A8
24 A9
23 A11
22 OE
21 A10
20 CE
19 I/O7
18 I/O6
17 I/O5
16 I/O4
15 I/O3

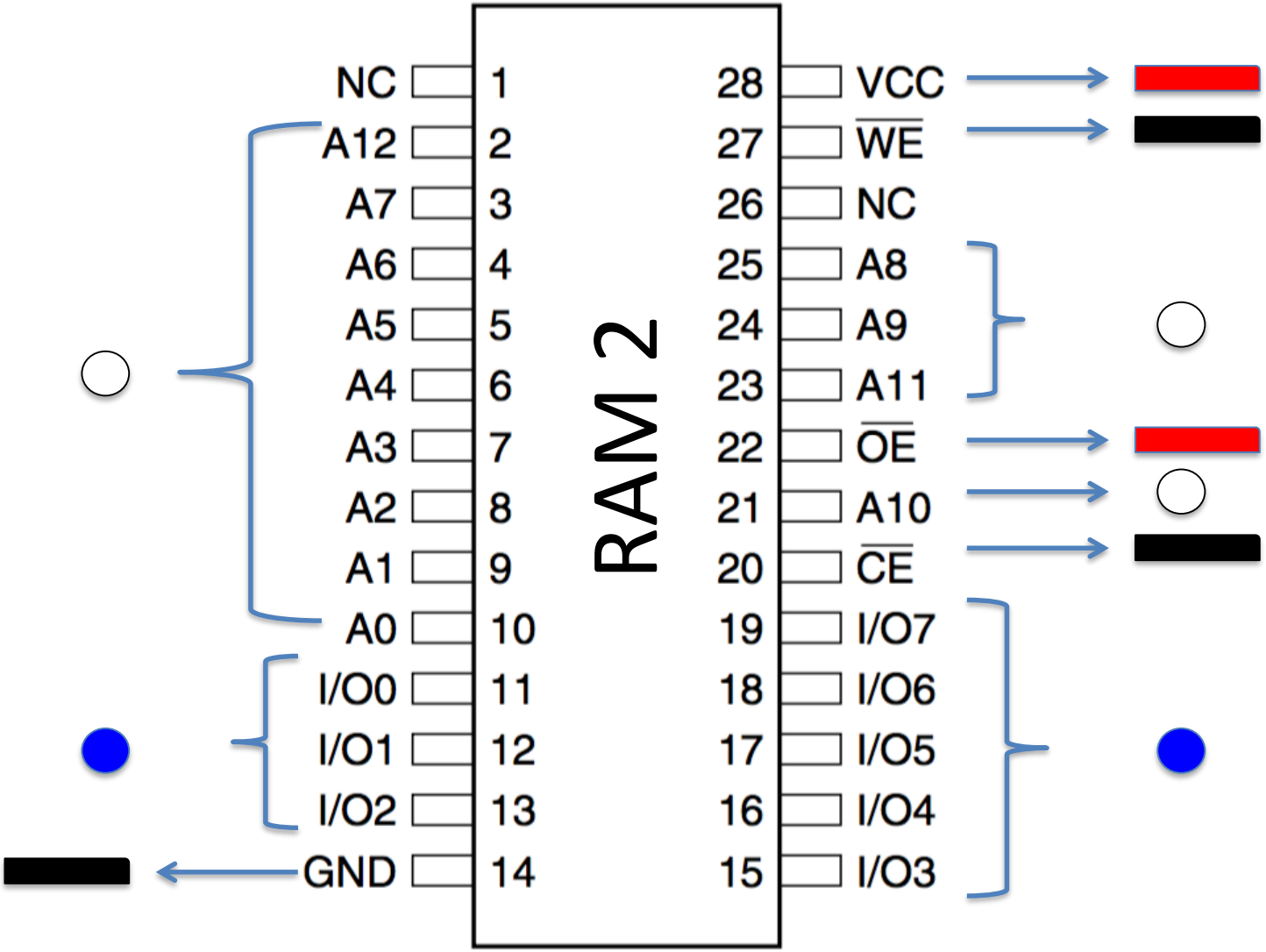
AT28C64B


3.3 V
NC 1
A12 2
A7 3
A6 4
A5 5
A4 6
A3 7
A2 8
A1 9
A0 10
I/O0 11
I/O1 12
I/O2 13
GND 14
28 VCC
27 WE
26 NC
25 A8
24 A9
23 A11
22 OE
21 A10
20 CE
19 I/O7
18 I/O6
17 I/O5
16 I/O4
15 I/O3
EEPROM 2 +S

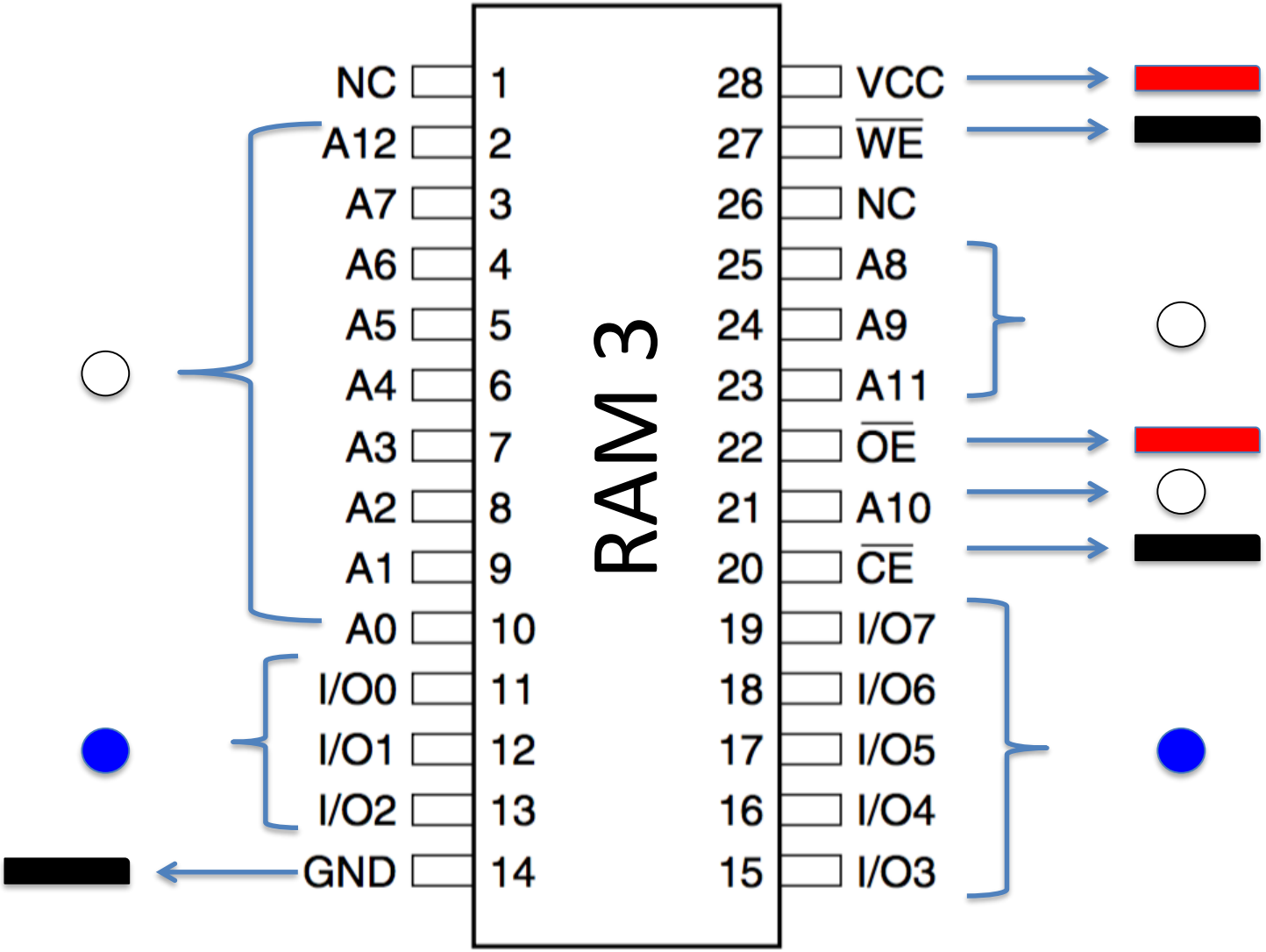
AT28C64B


NC 1
A12 2
A7 3
A6 4
A5 5
A4 6
A3 7
A2 8
A1 9
A0 10
I/O0 11
I/O1 12
I/O2 13
GND 14
28 VCC
27 WE
26 NC
25 A8
24 A9
23 A11
22 OE
21 A10
20 CE
19 I/O7
18 I/O6
17 I/O5
16 I/O4
15 I/O3
RAM1-S

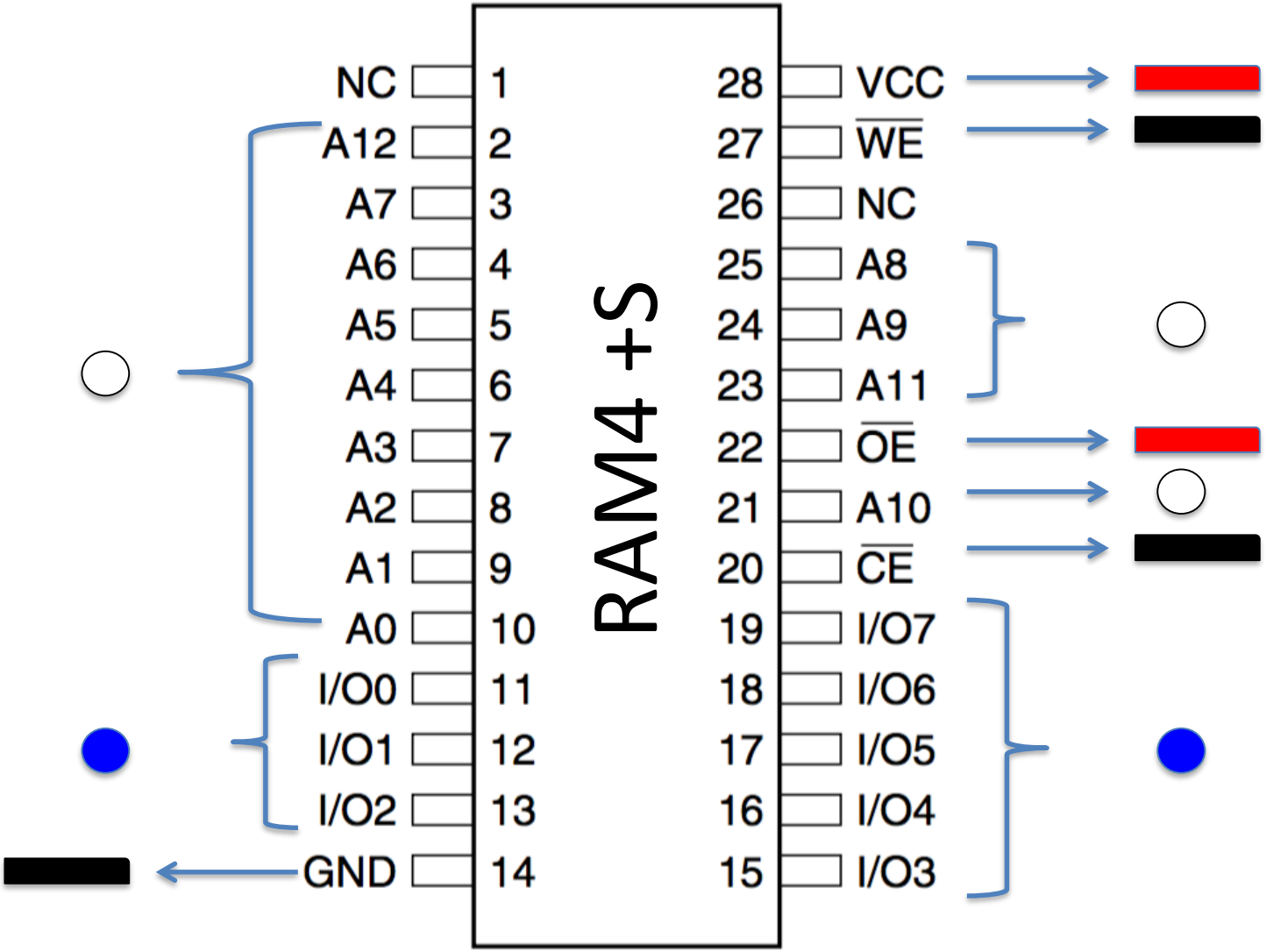
HY6264A


RAM 2
NC 1
A12 2
A7 3
A6 4
A5 5
A4 6
A3 7
A2 8
A1 9
A0 10
I/O0 11
I/O1 12
I/O2 13
GND 14
28 VCC
27 WE
26 NC
25 A8
24 A9
23 A11
22 OE
21 A10
20 CE
19 I/O7
18 I/O6
17 I/O5
16 I/O4
15 I/O3

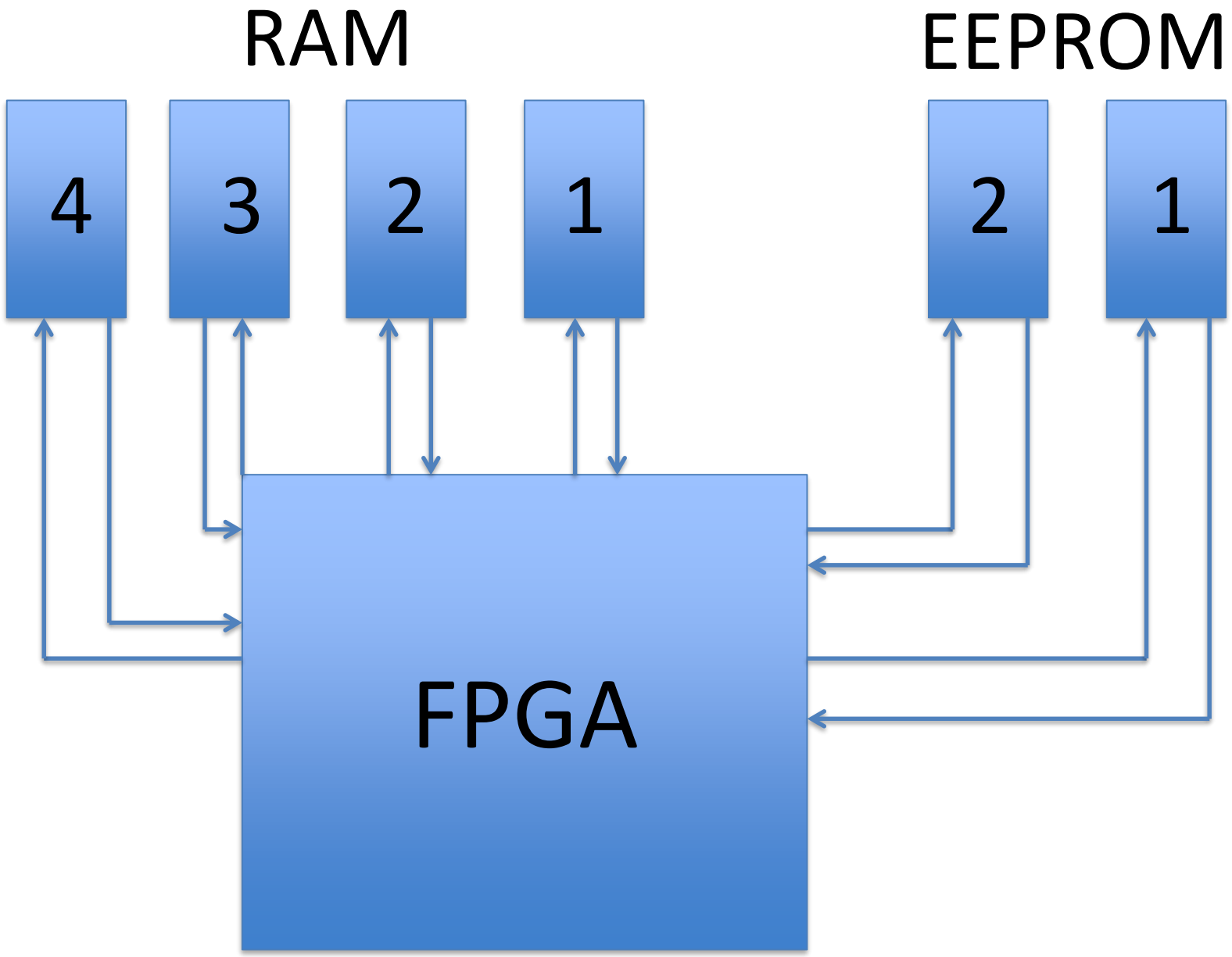
HY6264A


RAM 3
NC 1
A12 2
A7 3
A6 4
A5 5
A4 6
A3 7
A2 8
A1 9
A0 10
I/O0 11
I/O1 12
I/O2 13
GND 14
28 VCC
27 WE
26 NC
25 A8
24 A9
23 A11
22 OE
21 A10
20 CE
19 I/O7
18 I/O6
17 I/O5
16 I/O4
15 I/O3

HY6264A


NC 1
A12 2
A7 3
A6 4
A5 5
A4 6
A3 7
A2 8
A1 9
A0 10
I/O0 11
I/O1 12
I/O2 13
GND 14
28 VCC
27 WE
26 NC
25 A8
24 A9
23 A11
22 OE
21 A10
20 CE
19 I/O7
18 I/O6
17 I/O5
16 I/O4
15 I/O3
RAM4 +S

HY6264A


RAM
EEPROM
4
3
2
1
2
1
FPGA

CONTROL

STEPS

Aritmético-Lógicas

*ADC (reg) – Señales para leer del banco de registros (1), señales multiplexor para que salga el dato del banco (2), ALU Load (2), **ALU operator** (2), ZR_Load (3), ZR_Bus (4), Señales para escribir en Banco (5)

*ADD (imm) –– **T1**: Señales para leer del banco de registros, señales multiplexor para que salga el dato inmediato de la unidad de control, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

T2: No leer banco, Señales multiplexor para que salga el dato inmediato de la unidad de control, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

*ADD (reg) – **T1**: Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

T2: Señales para leer del banco de registros (dirección será de 4 bits) por lo que puede ser el PC, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* AND (reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* ASR(imm) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, señal otro multiplexor para que saque dato inmediato de 5 bits y se usarán solo esos 5 bits, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* ASR(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* BIC(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* EOR(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* LSL(imm) -- Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, señal otro multiplexor para que saque dato inmediato de 5 bits y se usarán solo esos 5 bits, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* LSL(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* LSR(imm) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, señal otro multiplexor para que saque dato inmediato de 5 bits y se usarán solo esos 5 bits, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* LSR(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* MUL – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* MVN(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* NOP – Nada

* ORR(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* REV – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* REV16 – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* REVSH – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* ROR (reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* RSB (imm) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco y el otro mux no importa lo que salga (o mandar un dato inmediato de puros ceros), ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* SBC (reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* SUB (imm)– – **T1**: Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, y en el otro multiplexor que salga dato inmediato, con sign extension en 00, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

T2: Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco y en el otro mux un dato inmediato, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* SUB (reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* SXTB – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* SXTH – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* TST(reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, **banderas a la unidad de control, de ahí ir a Fetch 1**

* UXTB – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* UXTH – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, Señales para escribir en Banco

* CMN (reg) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator (suma entre Rm y Rn), **banderas a la unidad de control, de ahí ir a Fetch 1**

* CMP (imm) – Señales para leer del banco de registros, señales multiplexor para que salga el dato del banco(Rn) y en el otro multiplexor un dato inmediato, ALU Load, ALU operator (resta Rn-imm8), **banderas a la unidad de control, de ahí ir a Fetch 1**

* CMP (reg) – **T1**: Señales para leer del banco de registros (Rn), señales multiplexor para que salga el dato del banco, ALU Load, ALU operator (Rn-Rm), **banderas a la unidad de control, de ahí ir a Fetch 1**

T2: Señales para leer del banco de registros (SP/Rm y Rn), señales multiplexor para que salga el dato del banco, ALU Load, ALU operator (Rn-Rm/SP?), **banderas a la unidad de control, de ahí ir a Fetch 1**

Aritméticas Especiales

* ADD (SP Plus Immediate)–**T1**: Leer SP del banco de registros, señales multiplexor para que salga el dato del banco y en el otro mux un dato inmediato, ALU Load, ALU operator, ZR_Load, ZR_Bus, escribir en Rd en el banco

T2: Leer SP del banco de registros, señales multiplexor para que salga el dato del banco y en el otro mux un dato inmediato de 7, ALU Load, ALU operator, ZR_Load, ZR_Bus, escribir en SP en el banco

* ADD (SP Plus Register)– Leer SP y Rd/Rm del banco de registros, señales multiplexor para que salga el dato del banco, ALU Load, ALU operator, ZR_Load, ZR_Bus, escribir en Rd en el banco

* ADR – Leer PC del banco de registros, señales multiplexor para que salga el dato del banco y en el otro el dato inmediato, ALU Load, ALU operator, ZR_Load, ZR_Bus, escribir en Rd en el banco

* SUB (SP Minus Immediate)– Leer SP del banco de registros, señales multiplexor para que salga el dato del banco y en el otro mux un dato inmediato de 7 bits, ALU Load, ALU operator, ZR_Load, ZR_Bus, escribir en Rd en el banco

Saltos

* B –

* BL –

* BLX –

* BX –

Transferencias

STORE

* STR (imm)– Leer Rn del banco de registros, señales multiplexor para que salga el dato del banco y en el otro un un inmediato, ALU Load, ALU operator(suma), ZR_Load, ZR_Bus, ARR_Load, ARR_Bus.
Mandando dato Rt al registro de dato y escribiéndolo en RAM: Leer dato del banco, paso por ALU o inout, DRR_BUS_Load, RAM_DDR_Bus.

* STR (reg) – Igual que STR (imm) pero lees Rn y Rm en banco de registros.

* STRB (imm) – Igual que STR (imm) pero guardas un byte de Rt en RAM

* STRB (reg) – Igual que STRB (imm) pero lees Rn y Rm en banco de registros.

* STRH (imm) – Igual que STRB (imm) pero guardas 16 bits de Rt en RAM

* STRH (reg) – Igual que STRB (imm) pero lees Rn y Rm en banco de registros.

LOAD

* LDR (imm) – Leer Rn del banco de registros, señales multiplexor para que salga el dato del banco y en el otro un un inmediato, ALU Load, ALU operator(suma), ZR_Load, ZR_Bus, ARR_Load, ARR_Bus.
Sacando lo que hay en esa dirección en RAM y escribiendo en banco: RAM_DRR_Bus, RAM_DRR_Load, DRR_Bus_bus, leer del data in en banco y escribirlo en Rt

* LDR (literal) – Igual que LDR (imm), pero lees PC del banco y lo sumas con un imm8

* LDR (reg) – Igual que LDR (imm), pero lees Rm y Rn del banco para sumarse

* LDRB (imm) – Igual que LDR (imm), pero sacas un byte de RAM, lo extiendes a 32 con ceros, y lo escribes en Rt

* LDRB (reg) – Igual que LDRB (imm), pero lees Rm y Rn en banco

* LDRH (imm) – Igual que LDRB (imm), pero sacas 16 bits de RAM

* LDRH (reg) – Igual que LDRB (reg), pero sacas 16 bits de RAM

* LDRSB (reg) – Igual que LDRB (reg), pero extiendes con signo el byte de RAM

* LDRSH (reg) – Igual que LDRSB (reg), pero extiendes con signo los 16 bits de RAM

Otras

* MOV (imm) – Leer dato inmediato de 8 bits extendido con ceros en multiplexor, Bypass en ALU, escribirlo en Rd

* MOV (reg) — **T1**: Leer Rm (4 bits) del banco de registros, pasarlo por multiplexor y Bypass en ALU, Bypass en ALU, escribirlo en Rd

T2: Igual que T1 pero RM es de 3 bits

** POP – saca un dato o varios de RAM y lo(s) escribe en Banco(y en PC)?
The lowest-numbered register is loaded from the lowest memory address, through to the highest-numbered register from the highest memory address.

** PUSH – guarda un registro o varios (y LR)del Banco en RAM?
The registers are stored in sequence, the lowest-numbered register to the lowest memory address, through to the highest-numbered register to the highest memory address.

OPERANDO ALU

```
resultado_suma_carry when "00000",                      -- ADC (Register)
resultado_suma  when "00001",      -- ADD (Imm 3-bit)/ADD(Reg)/ADD(SP Plus Imm)/ADD(SP Plus Reg)/ADR/CMN(reg)
resultado_and when "00010",                             -- AND (Register)
resultado_ASR when "00011",                             -- ASR (Immediate) / ASR (Register)
resultado_bit_clear when "00100",                       -- BIC (Register)
resultado_xor when "00101",                             -- EOR (Register)
resultado_LSL when "00110",                             -- LSL (Immediate) / LSL (Register)
resultado_LSR when "00111",                             -- LSR (Immediate) / LSR (Register)
"--------------------------------" when "01000",       -- MOV
resultado_mult(31 downto 0) when  "01001",              -- MUL
resultado_not when "01010",                             -- MVN (Register)
A when "01011",                                         -- NOP / Bypass
resultado_or when "01100",                              -- ORR (Register)
resultado_REV when "01101",                             -- REV
resultado_REV16 when "01110",                           -- REV16
resultado_REVSH when "01111",                           -- REVSH
resultado_rotate when "10000",                          -- ROR (Register)
resultado_not when "10001",                             -- RSB (Immediate)
resultado_Resta_carry when "10010",                     -- SBC (Register)
resultado_resta when "10011",          -- SUB (Imm 3-bit) /SUB (Reg) /CMP (imm) /CMP (reg) /SUB(SP Minus Imm)
resultado_SXTB when "10100",                            -- SXTB
resultado_SXTH when "10101",                            -- SXTH
"--------------------------------" when "10110",       -- TST (Register)
resultado_UXTB when "10111",                            -- UXTB
resultado_UXTH when "11000",                            -- UXTH
(others =>'0') when others;
```

FETCH

Máquinas de estado:

* Traer PC del Banco de Registros a registro AR por medio de un nuevo camino directo entre ellos, cargarlo (LOAD) en AR.

* Mandar dato en AR a la memoria de programa, incrementar PC en el banco de registros, esperar 4 ciclos de reloj a 25MHz para dar 160ns a memoria ROM física

* Después del tiempo de lectura, cargar la salida de la memoria ROM al IR en la Unidad de Control (IR_LOAD).

DECODE

* Decodificación IR (ver qué se va a hacer) y codificar (mandar señales individuales a todos lados)

EXECUTE

* Realizar Instrucción, ya sea de operación aritmética-lógica, transferencia (Load/Store), Saltos, etc., mandando señales de IR y otras de las Unidad de Control en sus respectivos tiempos de la máquina de estados.

* Al terminar reiniciar proceso (Fetch 1).