Micro Architecture Of ARM

Instructor: Dr. Vinicius Prado da Fonseca (vpradodafons@online.mun.ca)

MEMORIAL UNIVERSITY
www.mun.ca

Microarchitecture

- Microarchitecture is the connection between logic and architecture.
- Specific arrangement.
  - Registers
  - ALUs
  - Finite state machines (FSMs)
  - Memories
  - other logic building blocks (e.g. Multiplexers)

Microarchitecture

- A particular architecture, such as ARM, may have many different microarchitectures, each with different trade-offs of performance, cost, and complexity.
- They all run the same programs, (based on the same ISA) but their internal designs vary widely.

Microarchitecture

- Design an ARM CPU that implements:
    - Data processing instructions:
        - ADD
        - SUB
        - AND
        - ORR
    - Memory instructions:
        - LDR
        - STR
    - Unconditional branch:
        - B

Review of Fetch/Execute

The CPU controls the operations of a computer by fetching instructions indexed by the Program Counter (PC) from memory.

An instruction can cause the CPU to:

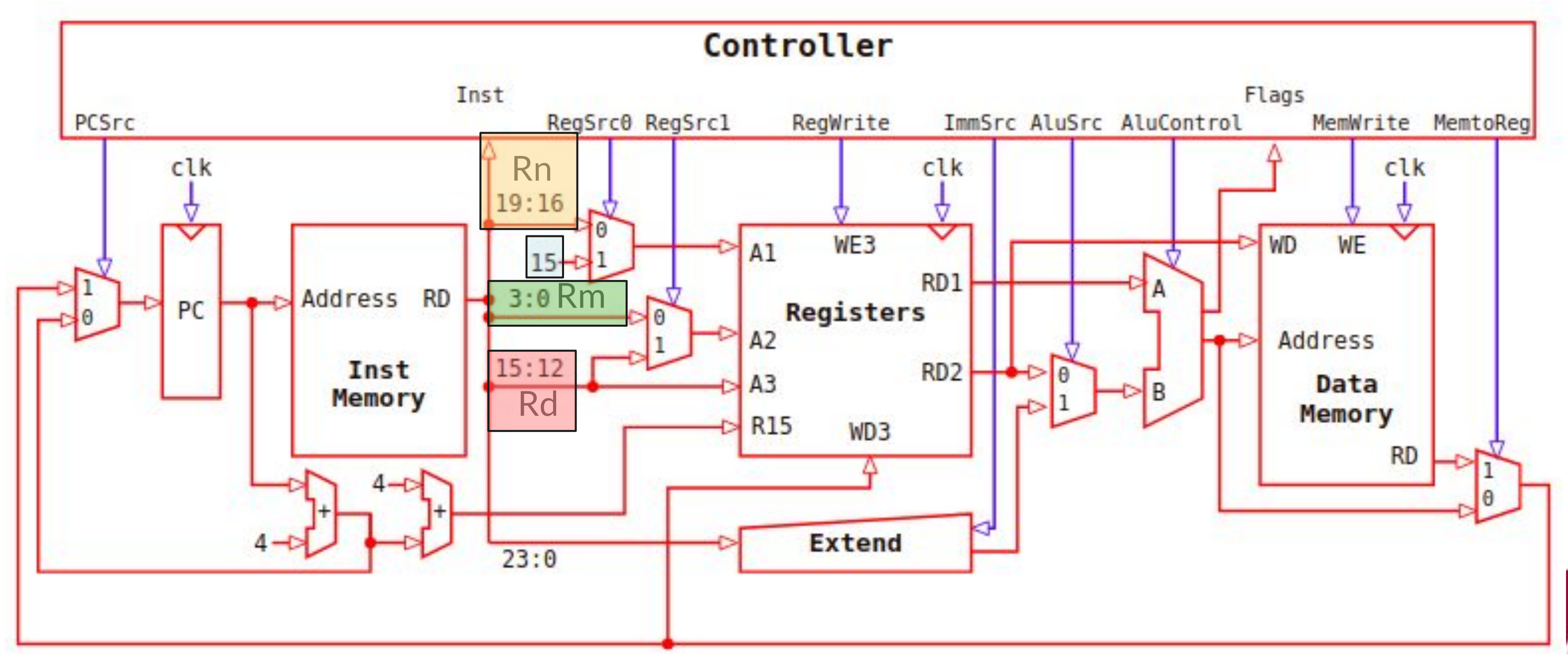
- Load data from memory
- Store data to memory
- Perform data operations on the register file (working storage)
- Conditional change the PC

Review of Fetch/Execute

- The fetch/execute algorithm that depicts the operations of a CPU
- The following single cycle CPU implements both the fetch and execute in one cycle
- One cycle execution requires two separate memories, one to provide the instruction and one for data
- All info I need is in the instruction itself
    - Controller splits the instruction

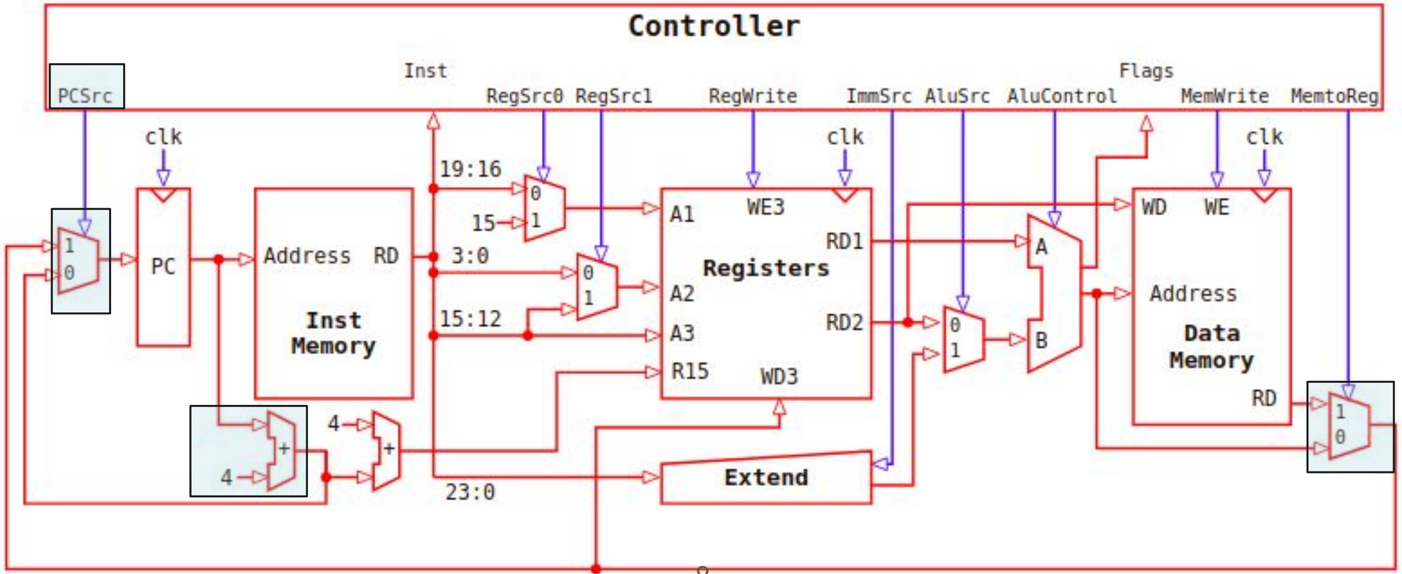
```
while the computer has power:

    IR <= Memory[PC]

    Decode and execute the
    instruction in the IR

    Adjust the PC

end while
```

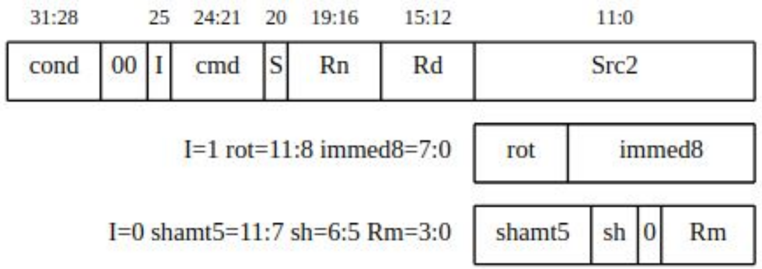
Control Lines



Control Lines - PCSrc

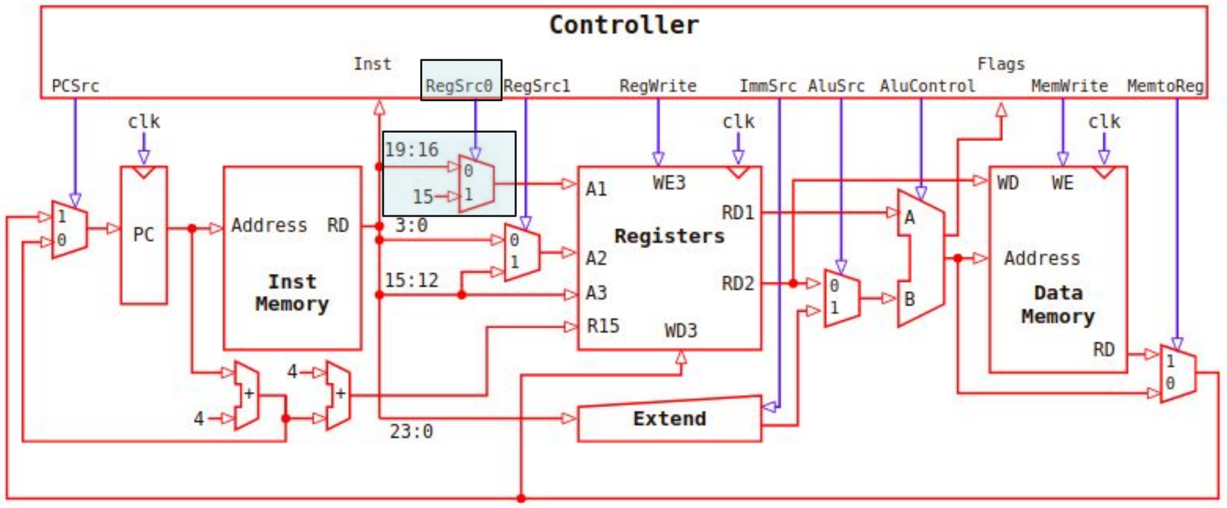
- Controls a multiplexor that selects the PC+4 when 0
- otherwise the output of the ALU is selected.
- Branch Example on slide 32

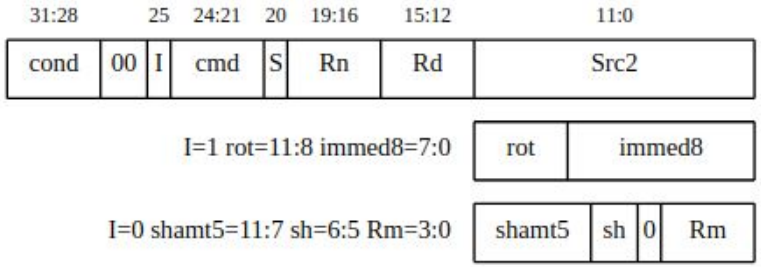




Control Lines - RegSrc0

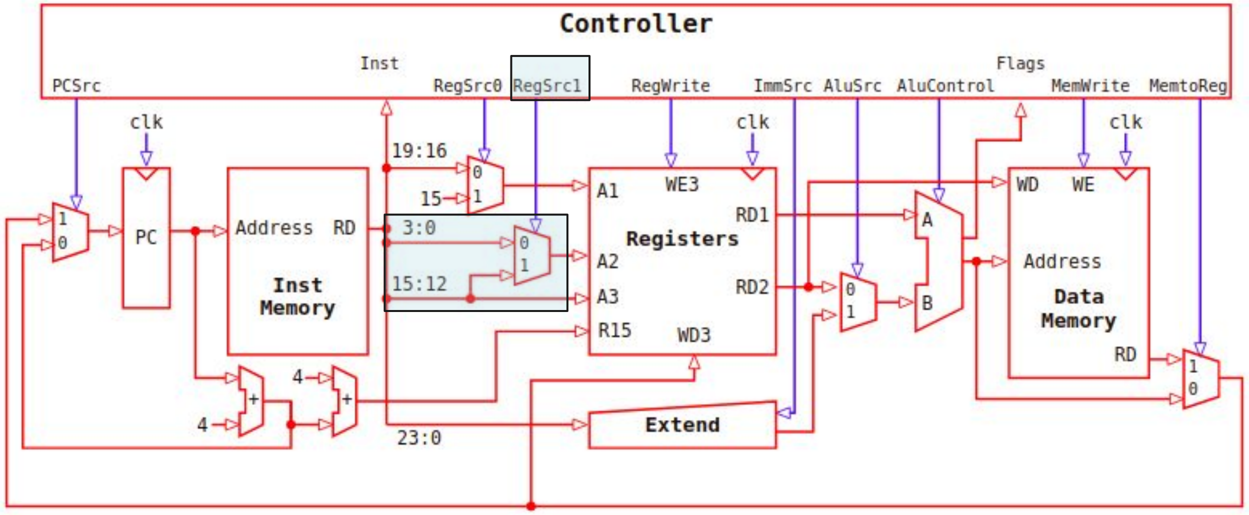
- Controls whether the PC (R15) is selected when 1
  - otherwise bits [19:16] are selected
- These bits select the Rn, the first operand of the ARM instructions or R15
- R15 is selected for the branch instruction

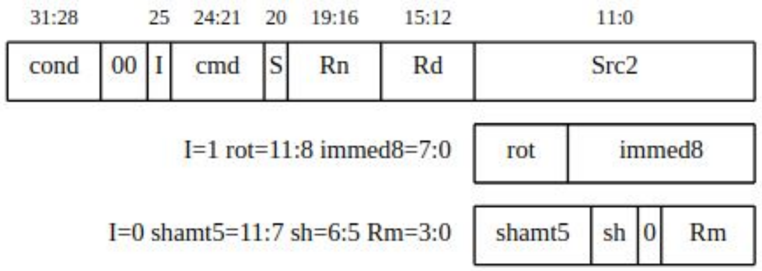




Control Lines - RegSrc1

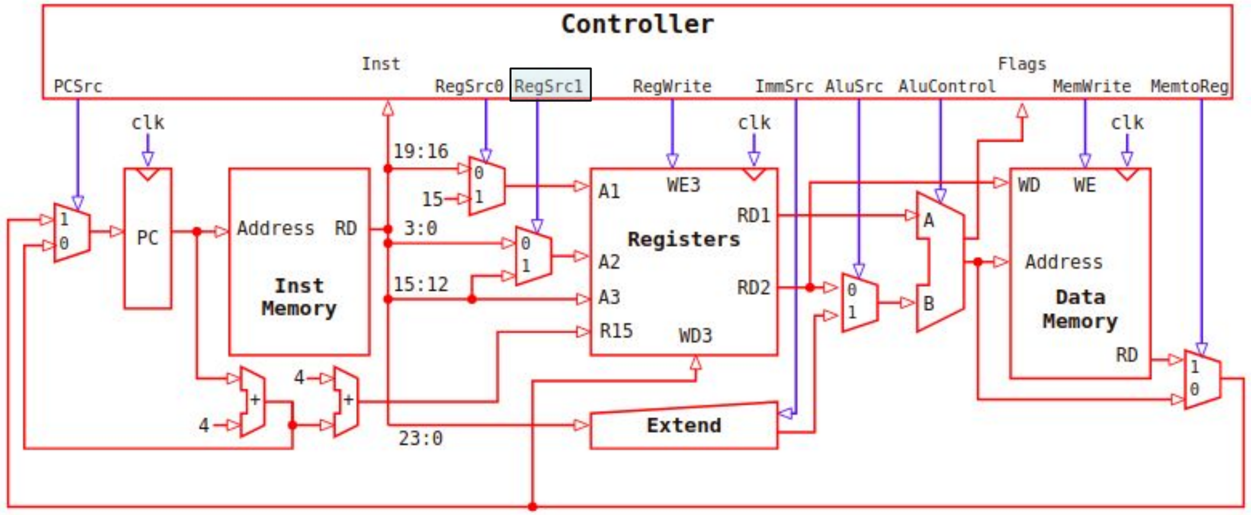
- Controls whether the Rd or Rm register is selected.
- The control is 0 when Rm is selected, otherwise Rd is selected (str instruction, format later).





Control Lines - RegSrc1

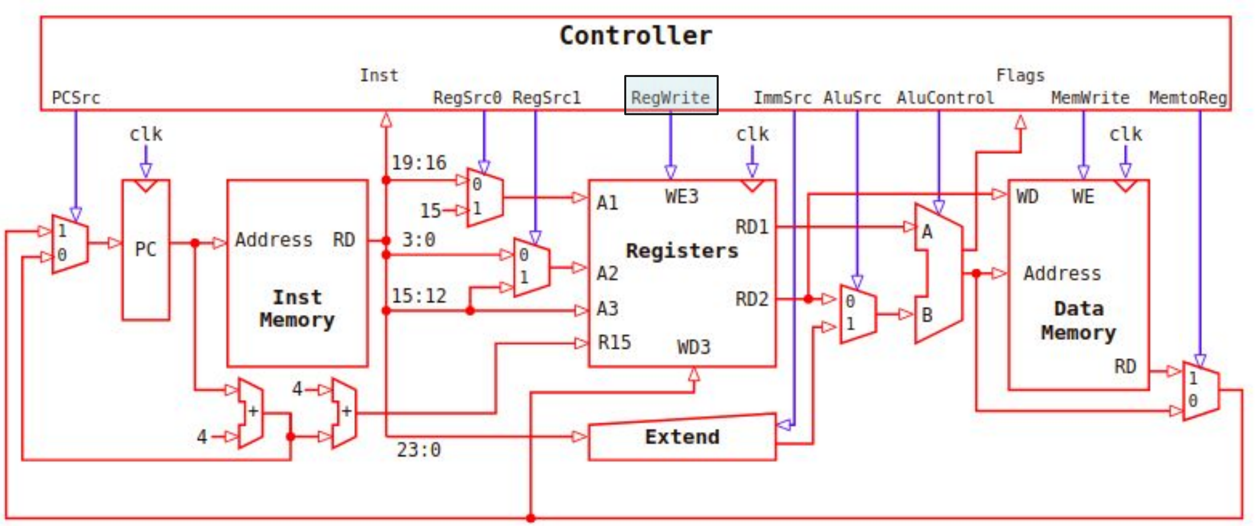
- The data processing instructions set control to 0.
- The store memory instructions sets the control to 1.
- Any instruction that uses immediate data is a do not care.



MEMORIAL UNIVERSITY
www.mun.ca

Control Lines - RegWrite

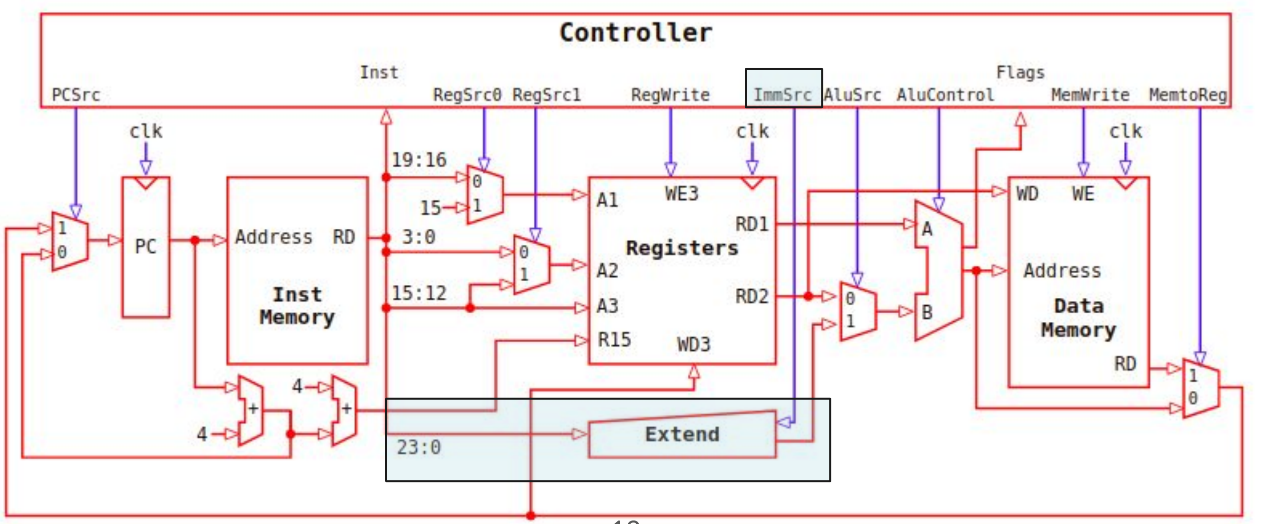
- The register file is updated with RegWrite is set to 1
  - otherwise no register is updated



MEMORIAL UNIVERSITY
www.mun.ca

Control Lines - ImmSrc

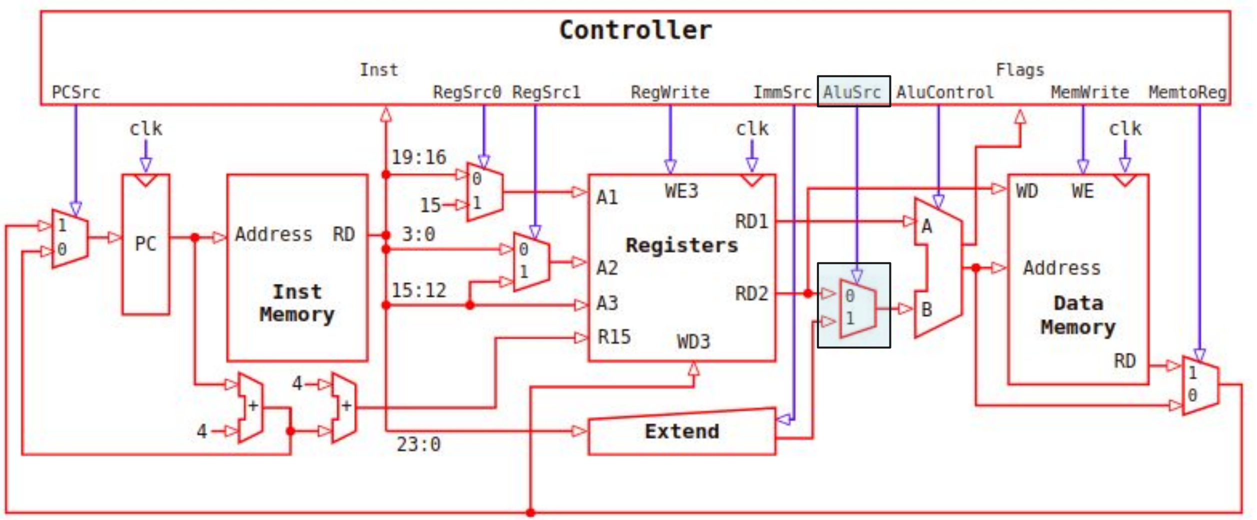
- Selects between zero extending bits [7:0] for the data processing instructions
- Zero extending bits [11:0] for the memory instructions
- Sign extending bits [23:0] and multiply by 4 for the branch instructions
- The controls are DATA, MEM and B



MEMORIAL UNIVERSITY
www.mun.ca

Control Lines - AluSrc

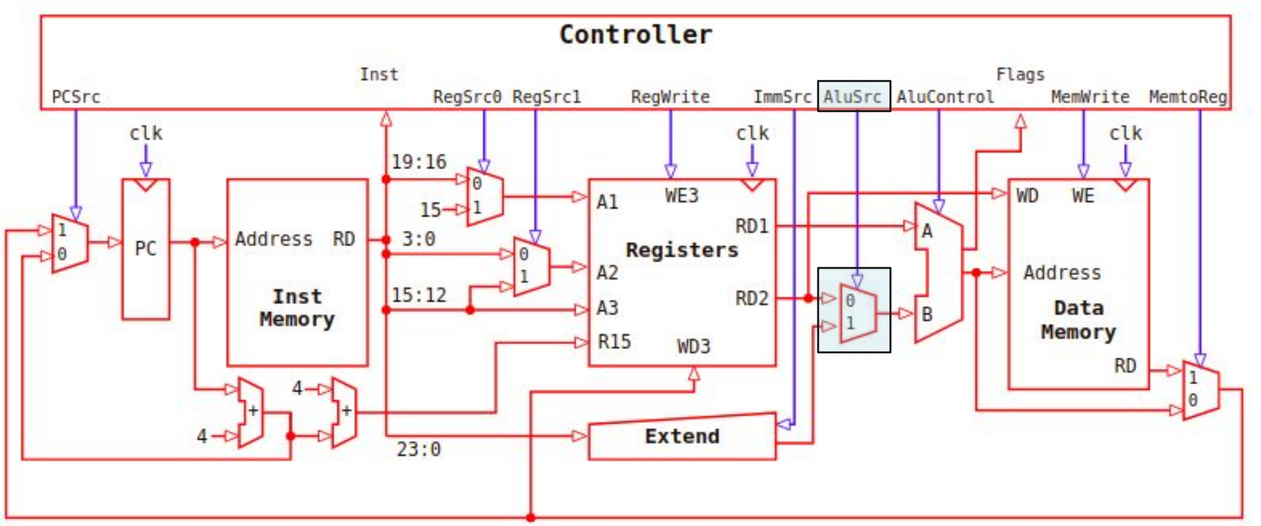
- Selects between read data port 2 (RD2)
  - or the resulted of the extend module
- The register is selected with the control is 0
  - otherwise the immediate extend module is selected



MEMORIAL UNIVERSITY
www.mun.ca

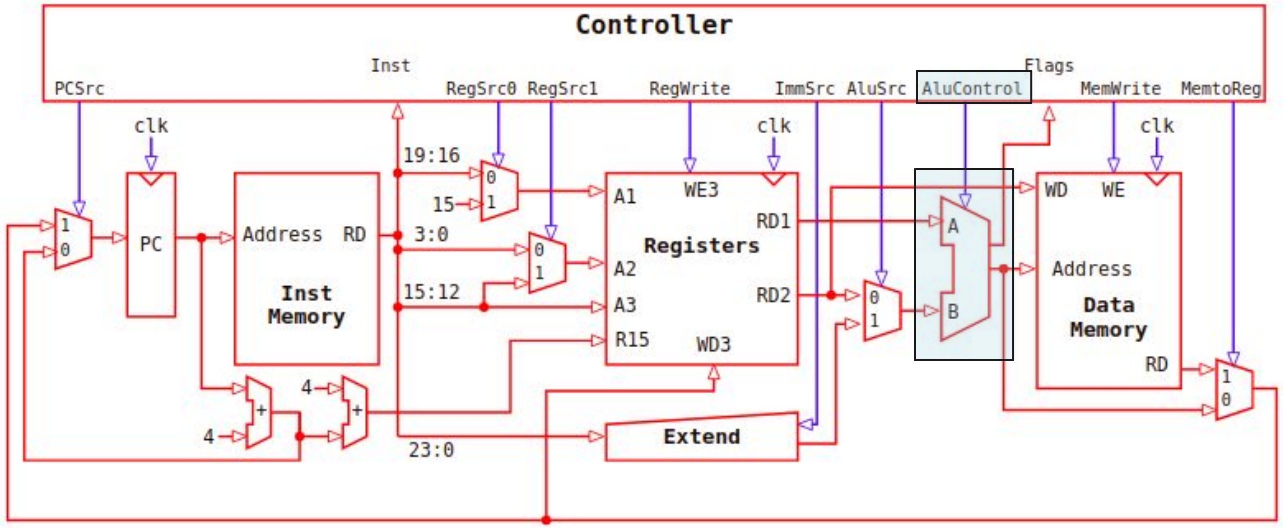
Control Lines - AluSrc

- The data processing instructions using only register operands sets control to 0
- The memory, branch, and data (using immediate) instructions sets the control to 1



Control Lines - AluControl

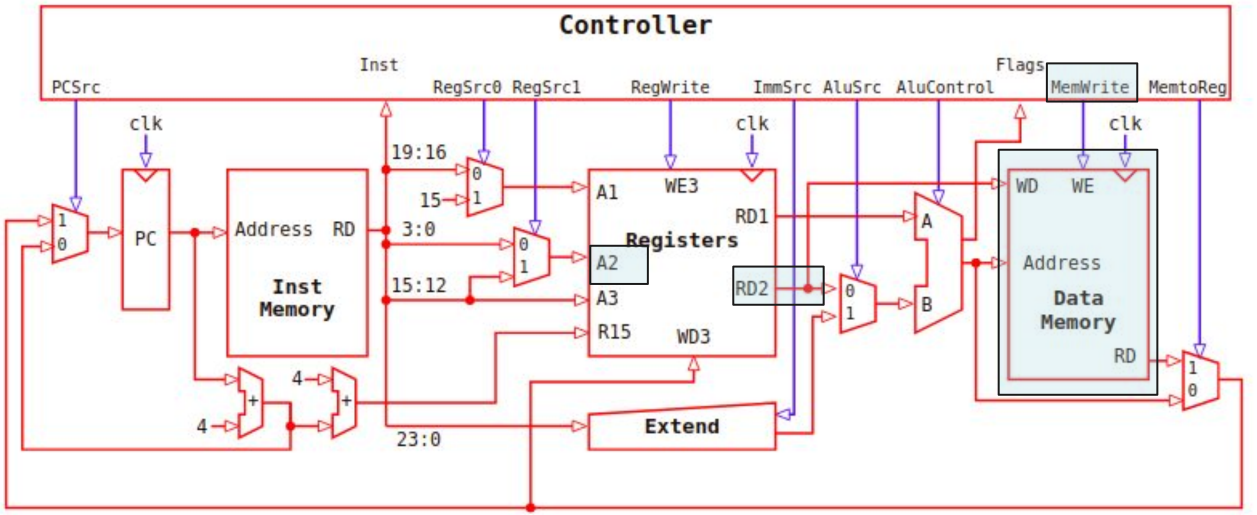
- The controls are: ADD, SUB, AND, OR



MEMORIAL UNIVERSITY
www.mun.ca

Control Lines - MemWrite

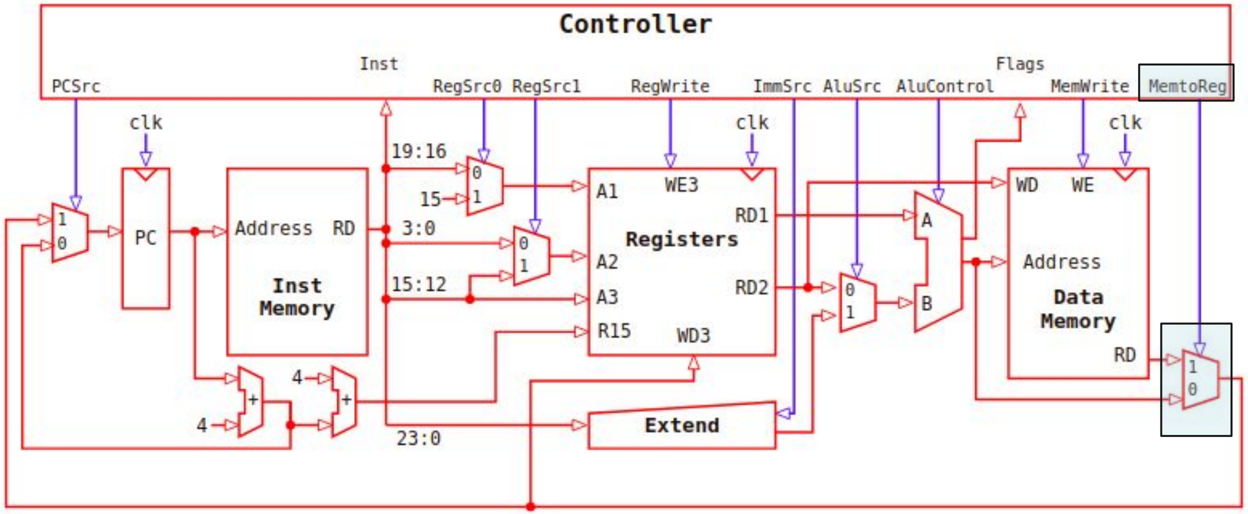
- Memory is updated with MemWrite is 1, data (WD) comes from RD2, RegSrc1 = 1
    - otherwise memory is not updated
- The STR instruction will set MemWrite to 1
- This control is 0 if the result is not used.



MEMORIAL UNIVERSITY
www.mun.ca

Control Lines - MemtoReg

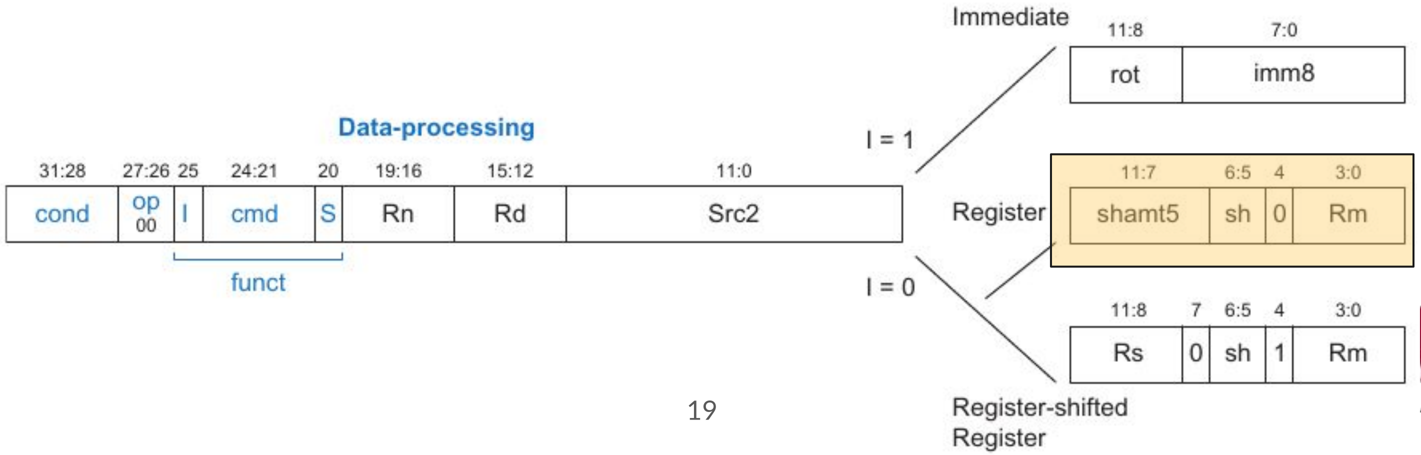
- This control selects between the data memory and the ALU
- MemtoReg is set to 1 for for LDR instructions, 0 for Data instructions (save to register), and dont care for STR instructions.



ORR R2, R3, R4 (R2 <= R3 | R4)

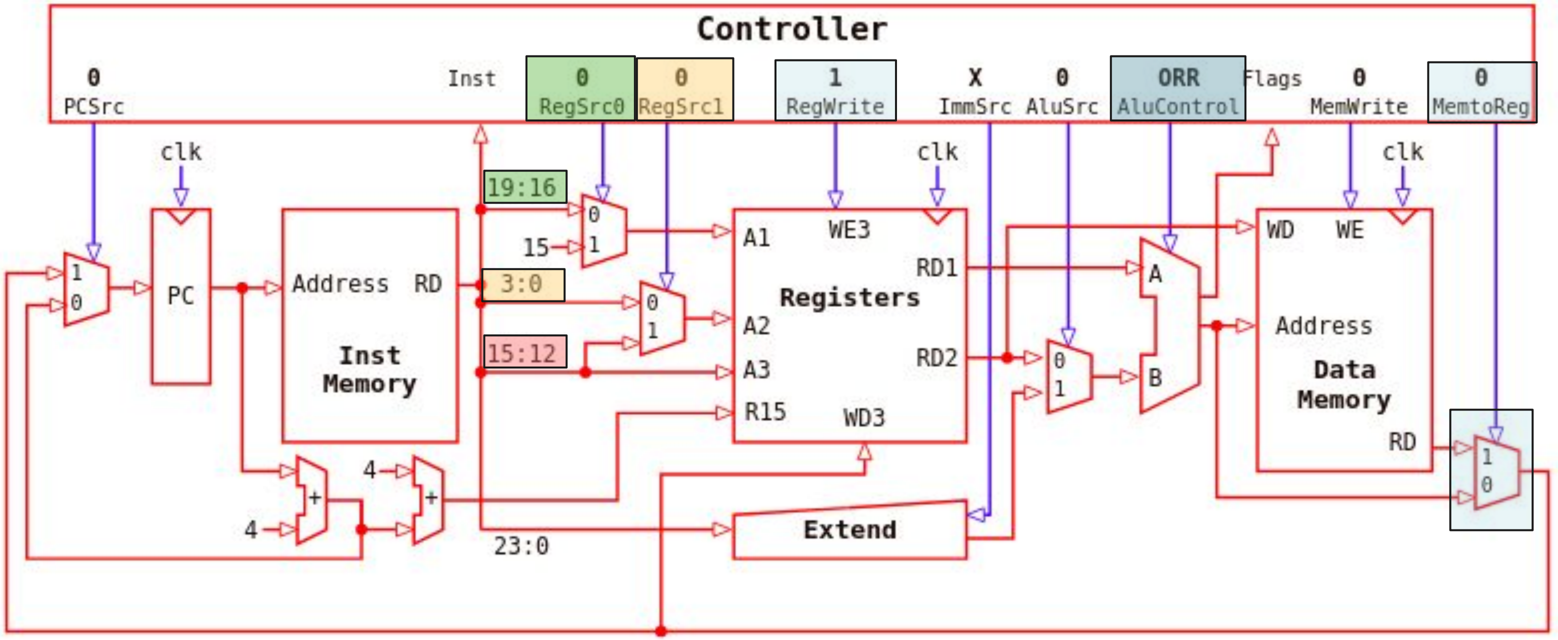
(e1832004) 1110 00 011000 0011 0010 00000 00 0 0100

cond	Op	I CMD S	Rn	Rd	shamt5 sh R Rm
1110	00	0 1100 0	0011	0010	00000 00 0 0100
31:28	27:26	25:20	19:16	15:12	11:0

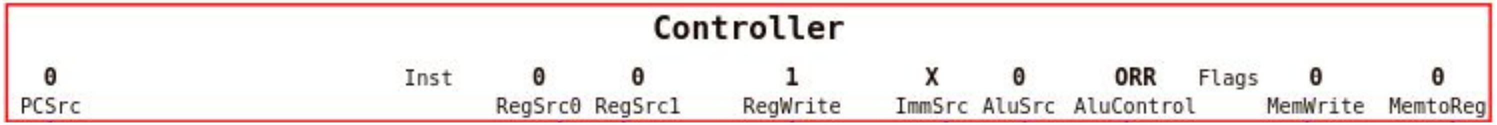


ORR R2, R3, R4 (R2 <= R3 | R4)

```
(e1832004) 1110 00 0 1100 0 0011 0010 00000 00 0 0100
```



ORR R2, R3, R4 (R2 <= R3 | R4)


Controller

PCSrc	Inst	RegSrc0	RegSrc1	RegWrite	ImmSrc	AluSrc	AluControl	Flags	MemWrite	MemtoReg
0		0	0	1	X	0	ORR		0	0

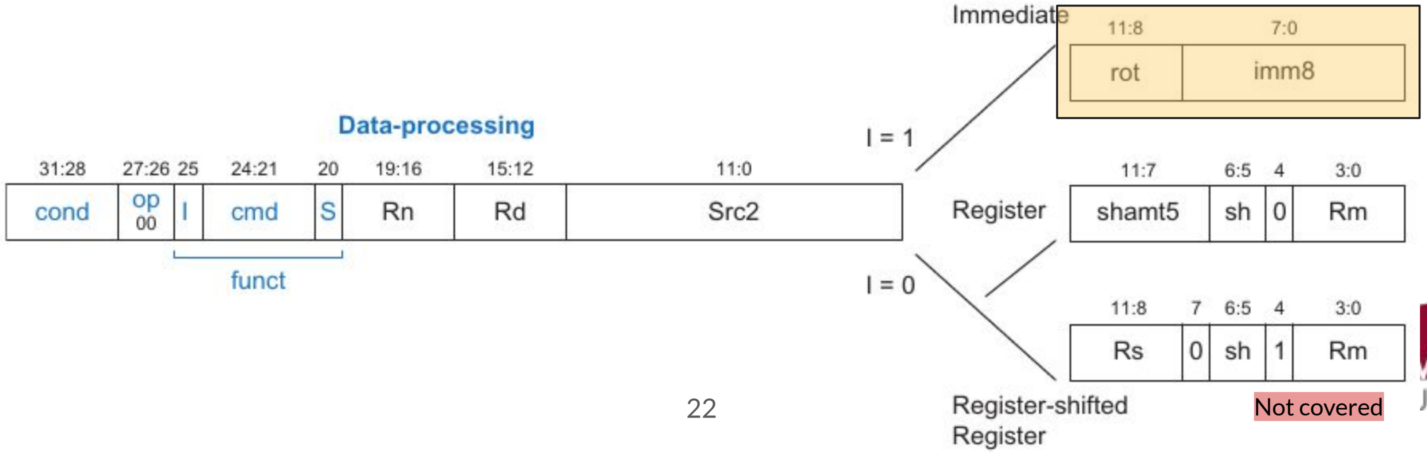
The rationale for these settings are:

- PCSrc=0 For all data processing instruction the next instruction (PC+4) is executed
- RegSrc0=0 The first source operand Rn is selected
- RegSrc1=0 The second source operand Rm is selected
- RegWrite=1 The result of ORR is stored, thus the register file is updated
- ImmSrc=X Since AluSrc=0 the output of the extend module is ignored
- AluSrc=0 Set to 0 to select data from the second register operand
- AluControl=ORR AluControl is ORR to OR the two registers
- MemWrite=0 The memory is not modified
- MemtoReg=0 The result from the ALU is sent to the register file

AND R3, R8, #15 (R3 <= R8 & 15) Immediate op [Left here]

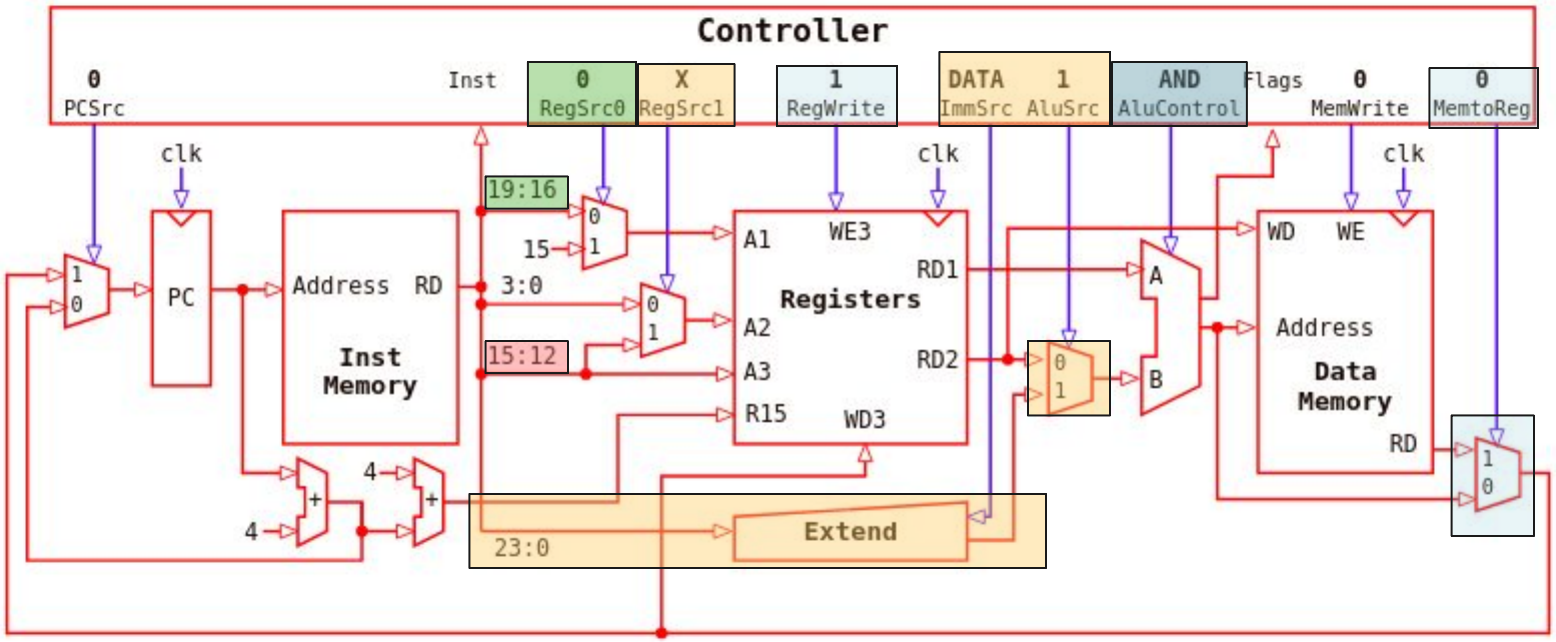
```
(e208300f) 1110 00 1 0000 0 1000 0011 0000 00001111
```

cond	Op	I CMD S	Rn	Rd	rot imm8
1110	00	1 0000 0	1000	0011	0000 00001111
31:28	27:26	25:20	19:16	15:12	11:0

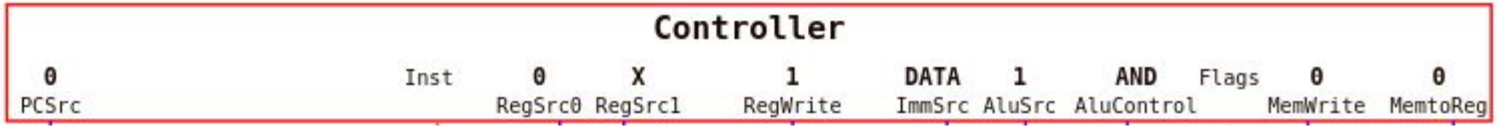


AND R3, R8, #15 (R3 <= R8 & 15) Immediate op

```
(e208300f) 1110 00 1 0000 0 1000 0011 0000 00001111
```



AND R3, R8, #15 (R3 <= R8 & 15) Immediate op



The rationale for these settings are:

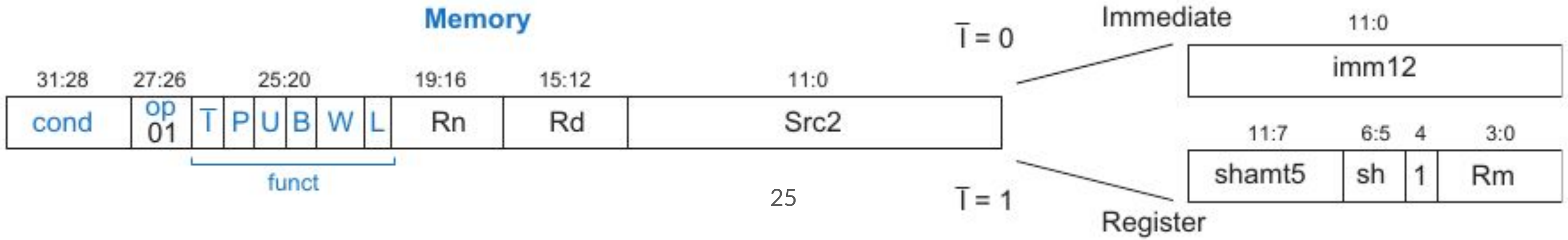
- PCSrc=0 For all data processing instruction the next instruction (PC+4) is executed.
- RegSrc0=0 The first source operand Rn is selected.
- RegSrc1=X The second source operand Rm is ignored.
- RegWrite=1 The result of AND is stored, thus the register file is updated.
- ImmSrc=DATA The immediate data operand is used, this zero extends the lower 8 bits.
- AluSrc=1 Set to 1 to select data from the extend module.
- AluControl=AND AluControl is AND to AND the the operands.
- MemWrite=0 The memory is not modified.
- MemtoReg=0 The result from the ALU is sent to the register file.

LDR R1, [R2] (R1 <= M[R2])

Transfers the content of memory specified by R2 to the R1 register (memory operation)
(Ī=immed;
PW = 10 = offset; // all instructions will use offset
LB: STR = 00; STRB = 01; LDR = 10, LDRB = 11;
U=1=ADD // all instruction will use add)

```
(e5921000) 1110 01 011001 0010 0001 000000000000
```

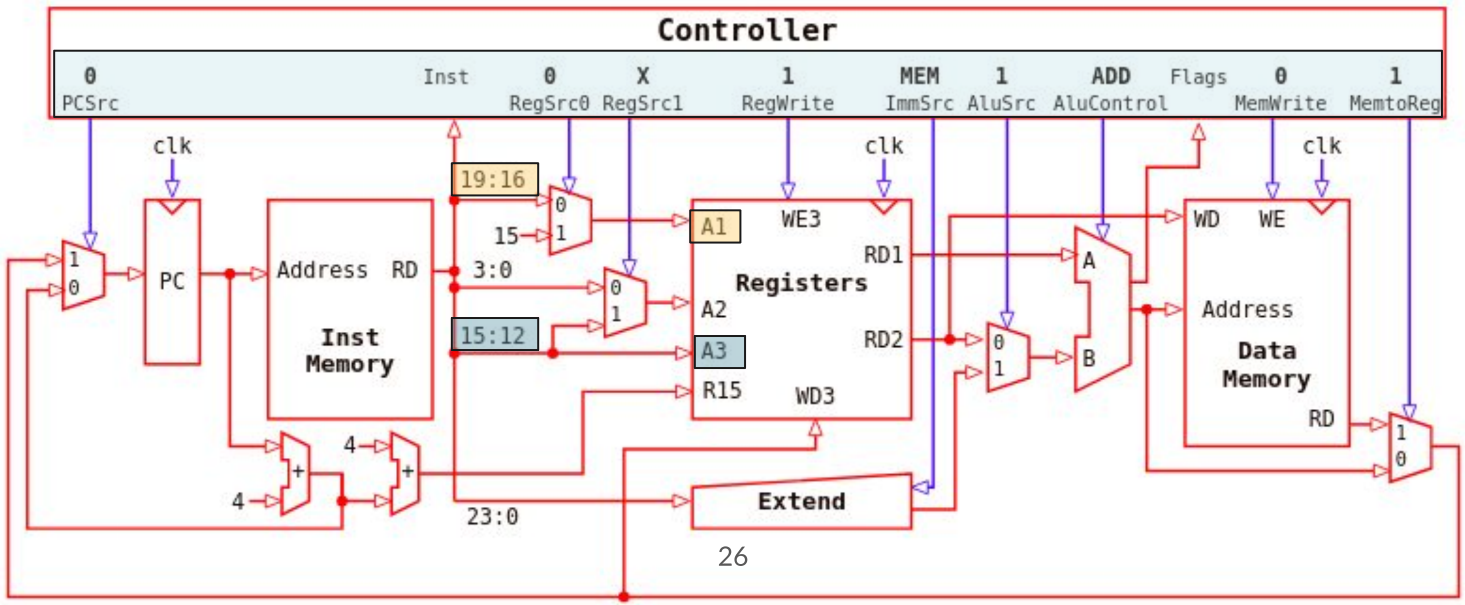
cond	Op	ĪPUBWL	Rn	Rd	Immd
1110	01	011001	0010	0001	000000000000
31:28	27:26	25:20	19:16	15:12	11:0



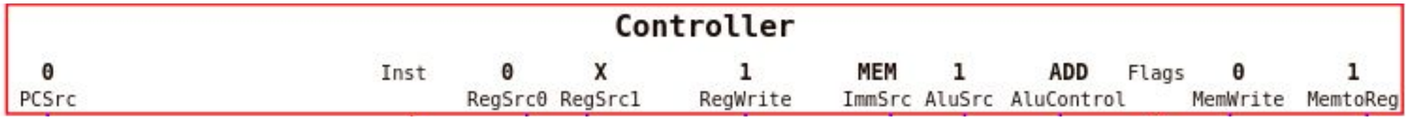
LDR R1, [R2] (R1 <= M[R2])

Transfers the content of memory specified by R2 to the R1 register

```
(e5921000) 1110 01 011001 0010 0001 000000000000
```



LDR R1, [R2] (R1 <= M[R2])



Controller

PCSrc	Inst	RegSrc0	RegSrc1	RegWrite	ImmSrc	AluSrc	AluControl	Flags	MemWrite	MemtoReg
0		0	X	1	MEM	1	ADD		0	1

The rationale for these settings are:

- PCSrc=0 In load instructions the next instruction will be executed; PC becomes PC+4
- RegSrc0=0 The Rn register is added to the immediate value to give the address of the location to read
- RegSrc1=X Since the read port 2 is not used by the LDR instruction; Rm not included
- RegWrite=1 The LDR instruction updates the register file, thus the control must be 1 to enable the write
- ImmSrc=MEM Selects bits 11:0 to calculate the address to be read; MEM op
  - These bits are zero extended
- AluSrc=1 Set to 1 to select data from the extend module
- AluControl=ADD AluControl is ADD to add the register and immediate value
- MemWrite=0 MemWrite is 0 to read data from the memory
- MemtoReg=1 MemtoReg is 1 to select data from the memory to send to the register file

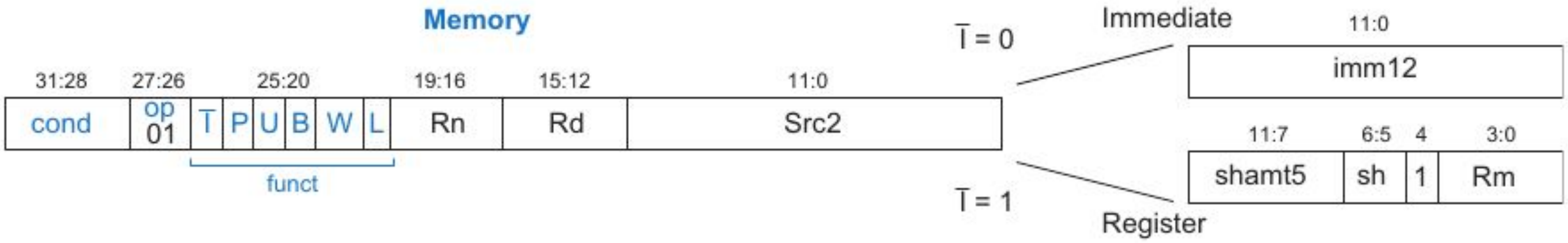
MEMORIAL UNIVERSITY
www.mun.ca

LDR R1, [R2, #40] (R1 <= M[R2 + 40])

Transfers the content of memory specified by R2 plus 40 to the R1 register

```
(e5921028) 1110 01 011001 0010 0001 000000101000 (40 is 0x28)
```

cond	Op	ĪPUBWL	Rn	Rd	Immd
1110	01	011001	0010	0001	000000101000
31:28	27:26	25:20	19:16	15:12	11:0



LDR R1, [R2, #40] (R1 <= M[R2 + 40])

Transfers the content of memory specified by R2 plus 40 to the R1 register

```
(e5921028) 1110 01 011001 0010 0001  000000101000  (40 is 0x28)
```

cond	Op	$\bar{I}$PUBWL	Rn	Rd	Immd
1110	01	011001	0010	0001	000000101000
31:28	27:26	25:20	19:16	15:12	11:0

In hardware the following are identical:

```
ldr r1, [r2]
ldr r1, [r2, #0]
```

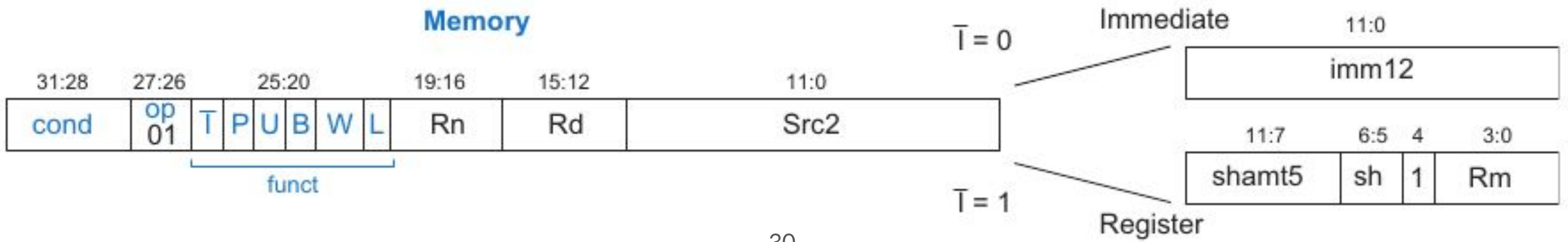
The assembler uses the simpler syntax when the offset is 0

STR R1,[R2,#48] (M[R2 + 48] <= R1)

The STR instruction copies R1 in to the location specified by R2 plus 48
(I=immed; PW = 10 = offset; LB => STR = 00; STRB = 01; LDR = 10; LDRB = 11; U=1=ADD)

```
(e5821030) 1110 01 011000 0010 0001 000000110000
```

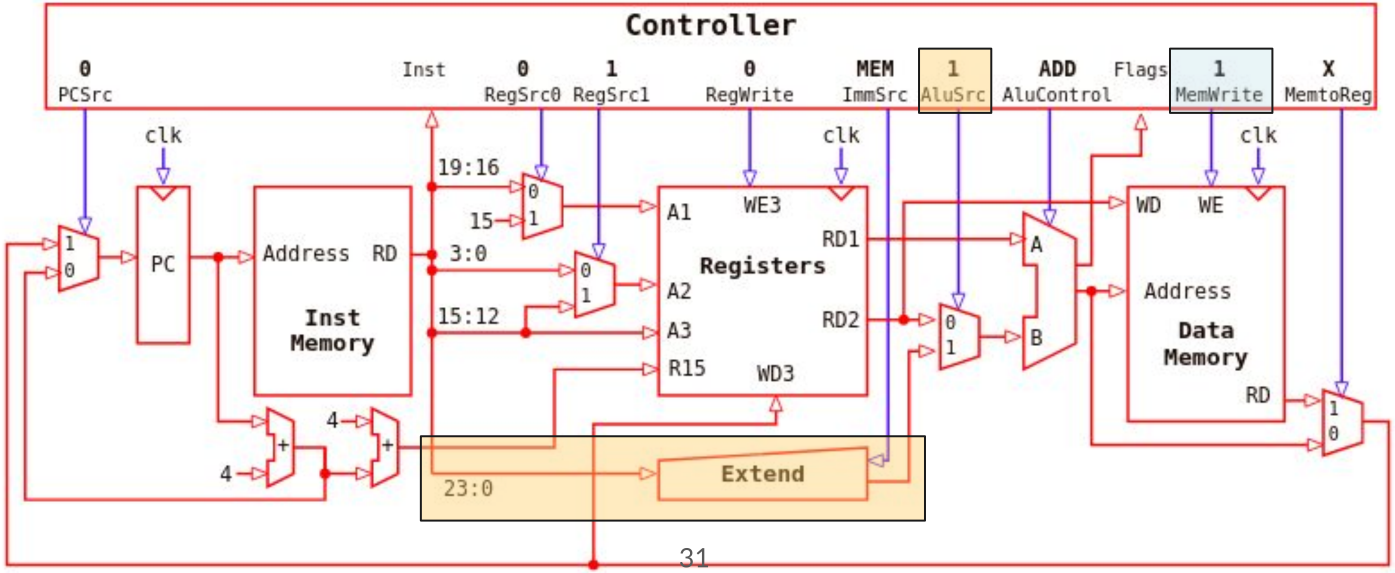
cond	Op	ĪPUBWL	Rn	Rd	Immd
1110	01	011000	0010	0001	000000110000
31:28	27:26	25:20	19:16	15:12	11:0



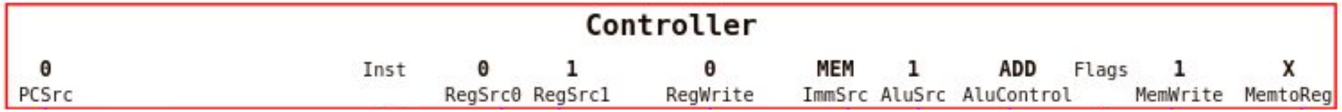
STR R1,[R2,#48] (M[R2 + 48] <= R1)

The STR instruction copies R1 in to the location specified by R2 plus 48

```
(e5821030) 1110 11 011000 0010 0001 000000110000
```



STR R1,[R2,#48] (M[R2 + 48] <= R1)



The rationale for these settings are:

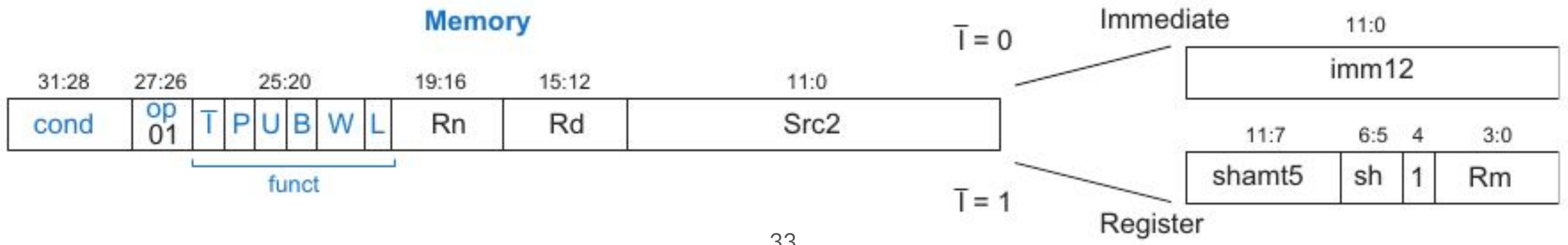
- PCSrc=0 In STR instructions the next instruction will be executed, so the PC becomes PC+4
- RegSrc0=0 The Rn register added to the immediate value to give the address of the location to read
- RegSrc1=1 Read port 2 outputs the register to be stored in memory; The register to store is specified in bits 15:12
- RegWrite=0 The register is not updated by the STR instruction
- ImmSrc=MEM Selects bits 11:0 to calculate the address to be read
- AluSrc=1 Set to 1 to select data from the extend module
- AluControl=ADD AluControl is ADD to add the register and immediate value
- MemWrite=1 MemWrite is 1 to write data to the memory
- MemtoReg=X MemtoReg is a don't care since the ALU result and data memory is not used

STR R1,[R2, R3, lsl #2] (M[R2 + R3 << 2] <= R1)

The STR instruction copies R1 in to the location specified by R2 plus R3 shifted by 2 (x4)
(Immediate Ī=1: shamt = 2 = 00010; sh = lsl = 00; RM = 3 =0011)

```
(e5821030) 1110 01 111000 0010 0001 00010 00 1 0011
```

cond	Op	ĪPUBWL	Rn	Rd	Register
1110	01	111000	0010	0001	00010 00 1 0011
31:28	27:26	25:20	19:16	15:12	11:0



Branch instructions

Branches to an address that is N bytes ahead (N/4 words) funct = (B = always 1; L = 1 = BL, 0 = B)
The 24-bit immediate field gives the number of instructions between target and PC+8
The extended module multiply branch offset by 4.
Example ARM Assembly Code

```
  0x80A0              B THERE                // branch
  0x80A4              ADDEQ R0, R1, R2       // PC+4
  0x80A8              SUB R0, R0, R9     // PC+8
 [0x80AC              ADD SP, SP, #8
2[0x80B0 THERE        SUB R0, R0, #1     //PC+8+(2 instructions)
  0x80B4              MOV PC, LR
  0x80B8              ADD R3, R3, #0x5

1011 10 10 0000 0000 0000 0000 0000 0010
```

immediate is 2 because target is three instructions past PC+8

B +2

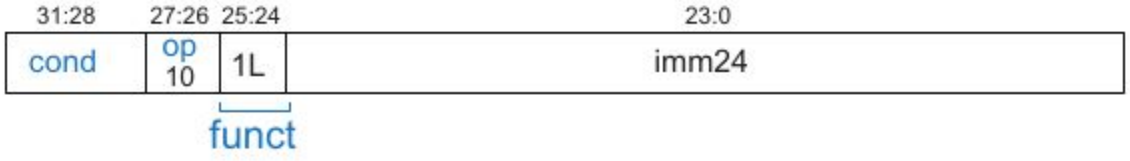
Branches to an address that is 2 words/instructions ahead of PC+8 (16 bytes of PC)

funct = (B = always = 1; L = 1 = BL, 0 = B)

```
(ea000002) 1110 10 10 000000000000000000000010
```

cond	Op	1L	imm24
1110	10	10	000000000000000000000010
31:28	27:26	25:24	23:0

Branch

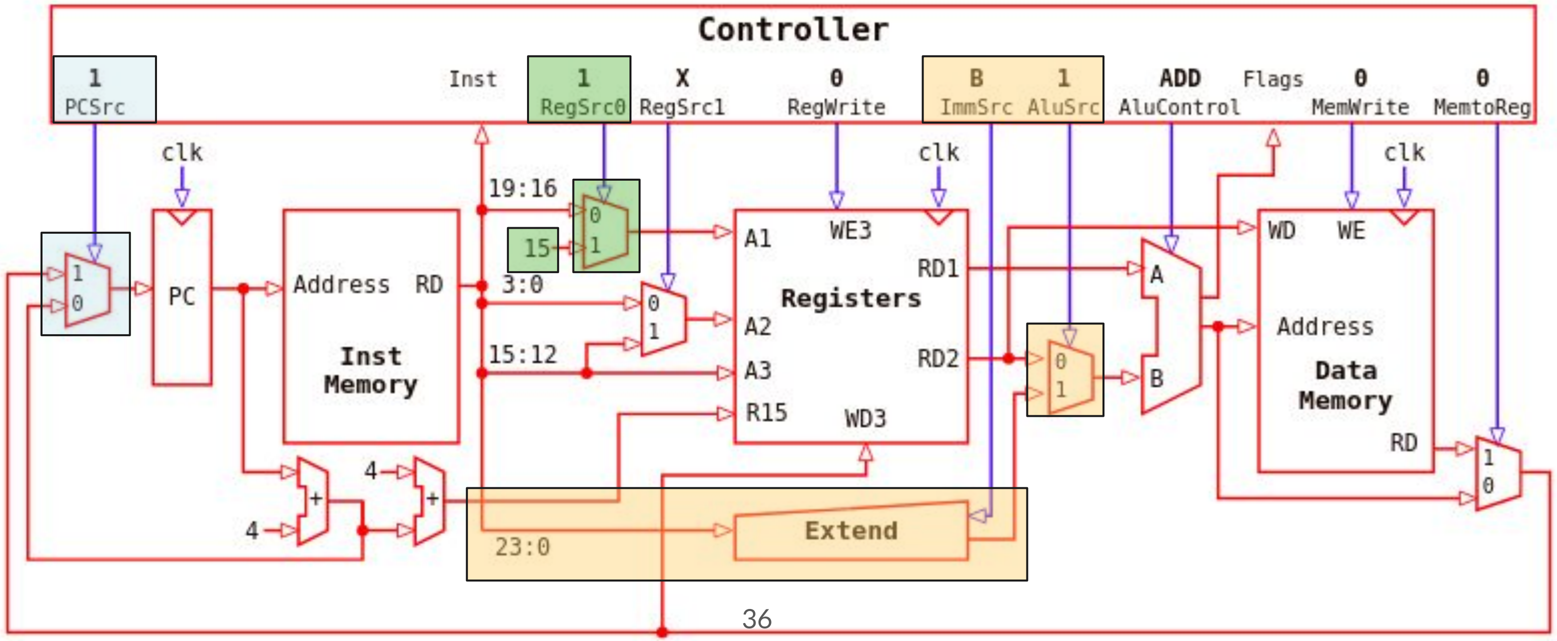


MEMORIAL UNIVERSITY
www.mun.ca

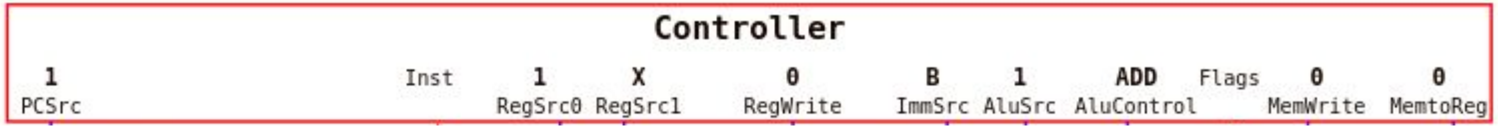
B +2

Branches to an address that is 16 bytes ahead (PC + 8 + (2*4))

```
(ea000002) 1110 10 10 000000000000000000000010
```



B +2



The rationale for these settings are:

- PCSrc=1 The target of the branch address is sent to the PC.
- RegSrc0=1 The R15 (PC) is selected.
- RegSrc1=X The second source operand Rm is ignored.
- RegWrite=0 No registers are updated.
- ImmSrc=B The lower 23 bits are sign extended and multiplied by 4 to be added to the PC.
- AluSrc=1 Set to 1 to select data from the extend module.
- AluControl=ADD AluControl is ADD to add the the operands.
- MemWrite=0 The memory is not modified.
- MemtoReg=0 The result from the ALU is sent to the PC.

Recommended

- Review instructions format (Sections 6.4)
- Read section 7.3 textbook (Single-cycle Processor)

That’s all folks

Final exam info soon