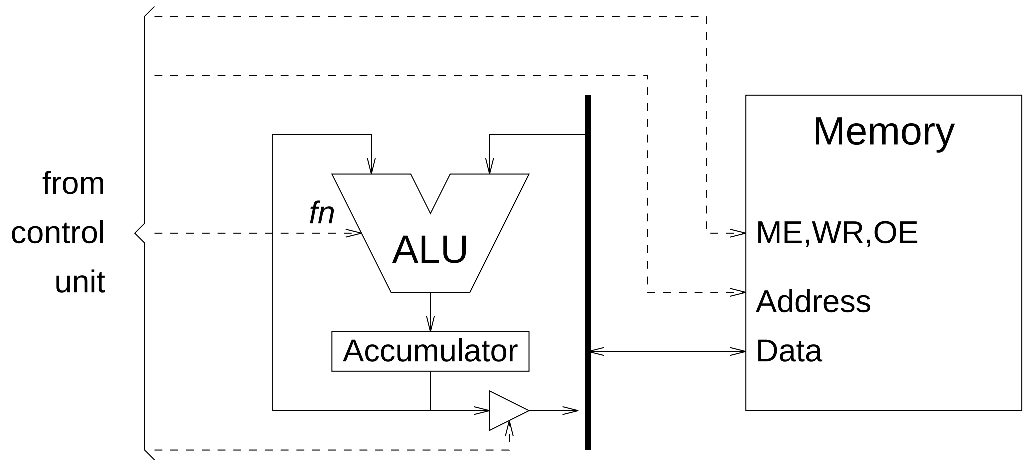
Datapath

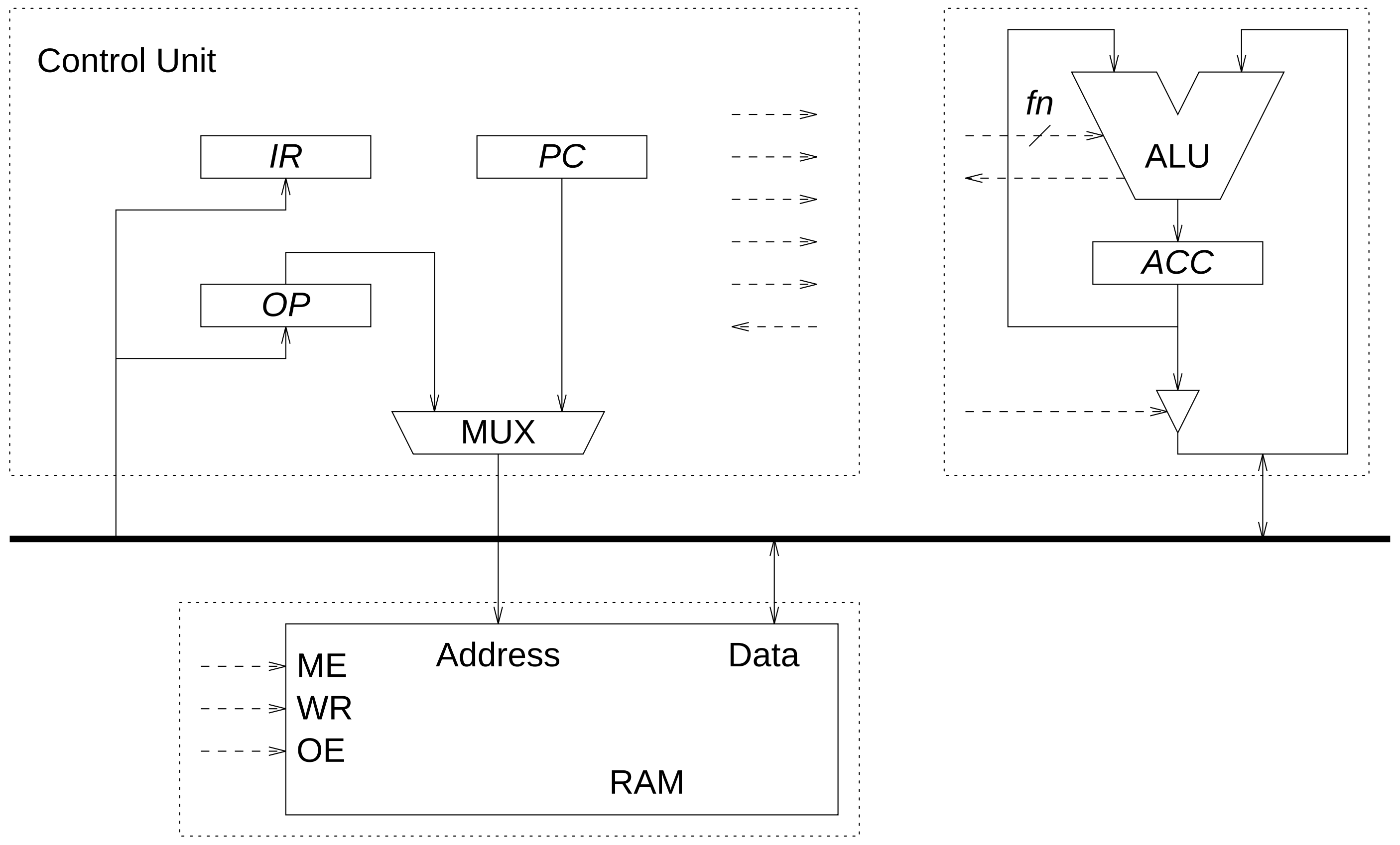
- Bus Based Architecture



Since wiring on a chip is expensive (requires a lot of area) we wish to limit the number of busses which need to be routed. A single data bus will be used wherever possible.

Microprocessor

- Including Control Unit



Microprocessor

- Instruction Cycle
  - ALU operation

1	$IR' \leftarrow mem(PC)$	$PC' \leftarrow PC + 1$
2	$OP' \leftarrow mem(PC)$	$PC' \leftarrow PC + 1$
3	$Acc' \leftarrow \mathcal{F}\{Acc, mem(OP)\}$	

Each stage includes one memory access.

- For synchronous single cycle memory[1], the complete instruction cycle takes 3 clock cycles.
- For multi cycle memory we may expect the instruction cycle to take 9 or more cycles.

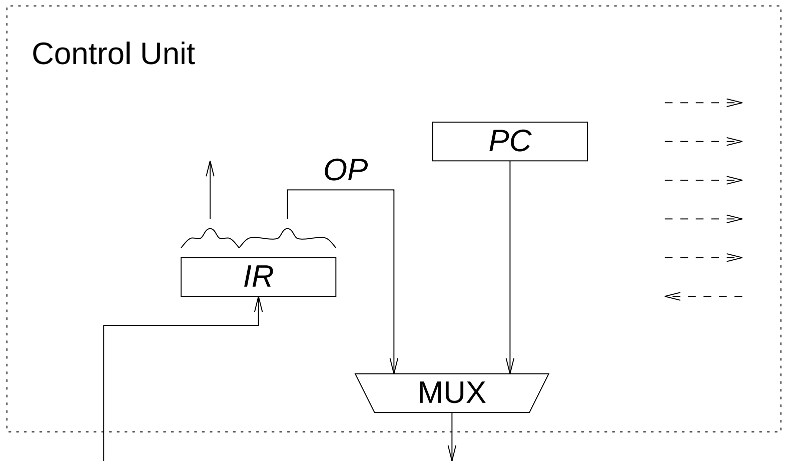
  Each memory access will include address setup, data access & data hold cycles.

[1]i.e. $ME = \overline{CLK}$

Microprocessor

Faster access is possible where the instruction register is large enough to include operand information as well as instruction information.

- Modified Control Unit



Microprocessor

- Instruction Cycles
  - ALU operation

1	$IR' \leftarrow mem(PC)$	$PC' \leftarrow PC + 1$
2	$Acc' \leftarrow \mathcal{F}\{Acc, mem(OP)\}$	

  - Store Operation

1	$IR' \leftarrow mem(PC)$	$PC' \leftarrow PC + 1$
2	$mem(OP)' \leftarrow Acc$	

  - Branch Operation

- - taken

1	$IR' \leftarrow mem(PC)$	$PC' \leftarrow PC + 1$
2	$PC' \leftarrow PC + OP$	

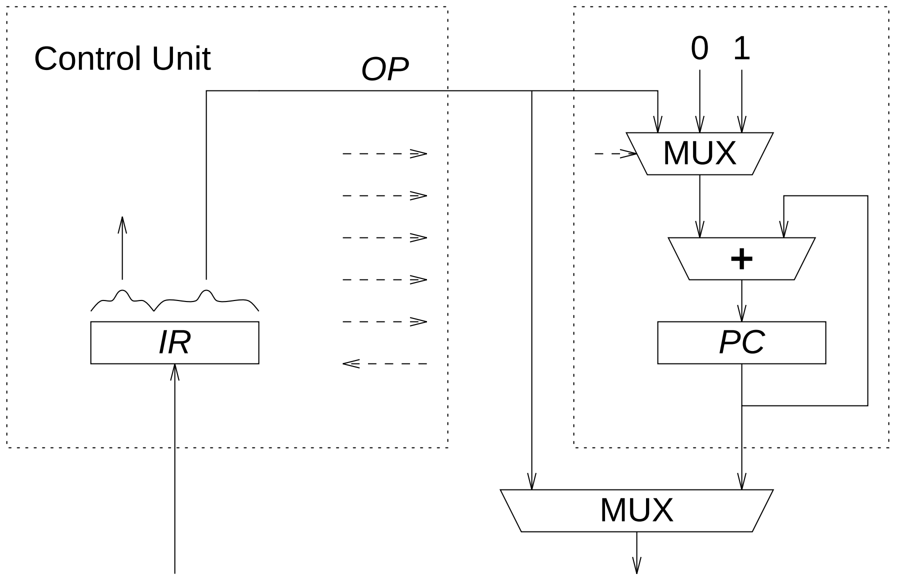
- - not taken

1	$IR' \leftarrow mem(PC)$	$PC' \leftarrow PC + 1$

Microprocessor

- Program Counter Unit

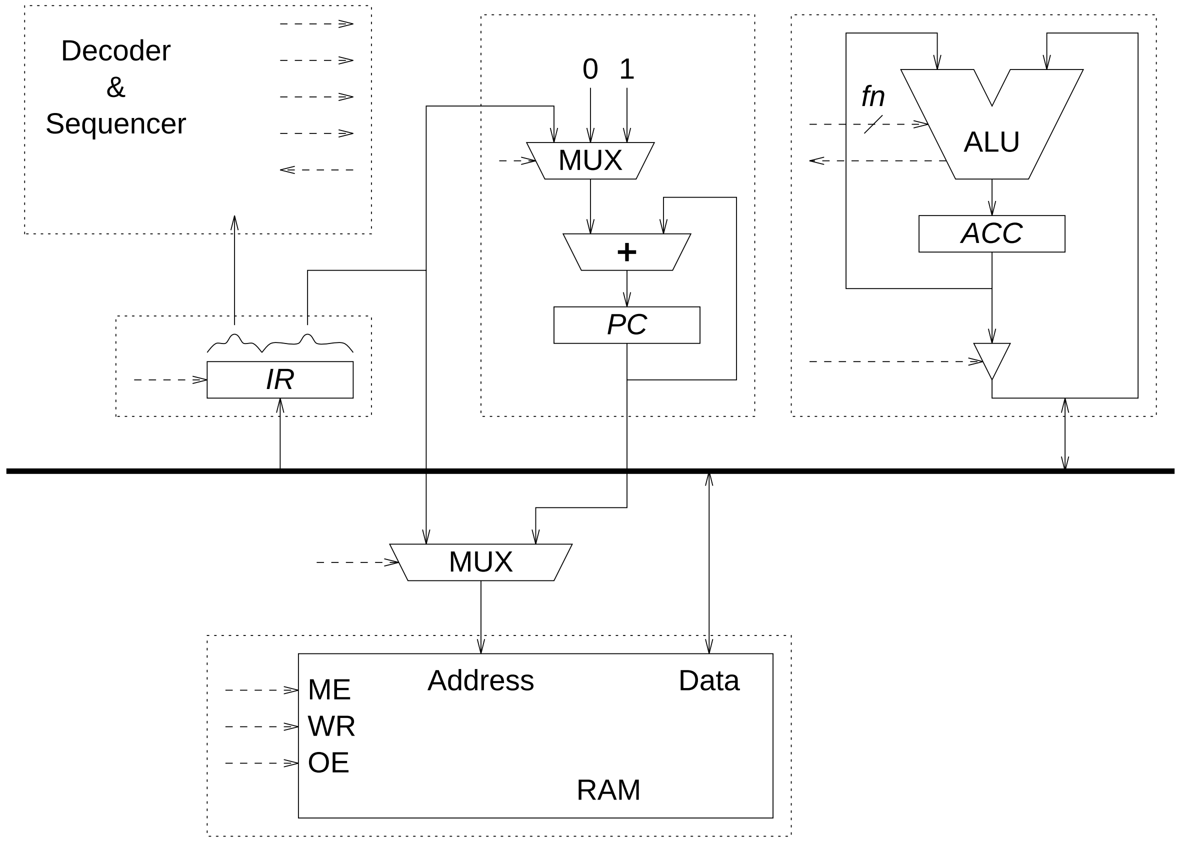
We can continue our top down approach by completing the design of a program counter unit:



In each cycle we add 0 or 1 or OP to the PC.

Microprocessor

- Complete Architecture



Microprocessor

- Complete Architecture
  - Well defined functional blocks

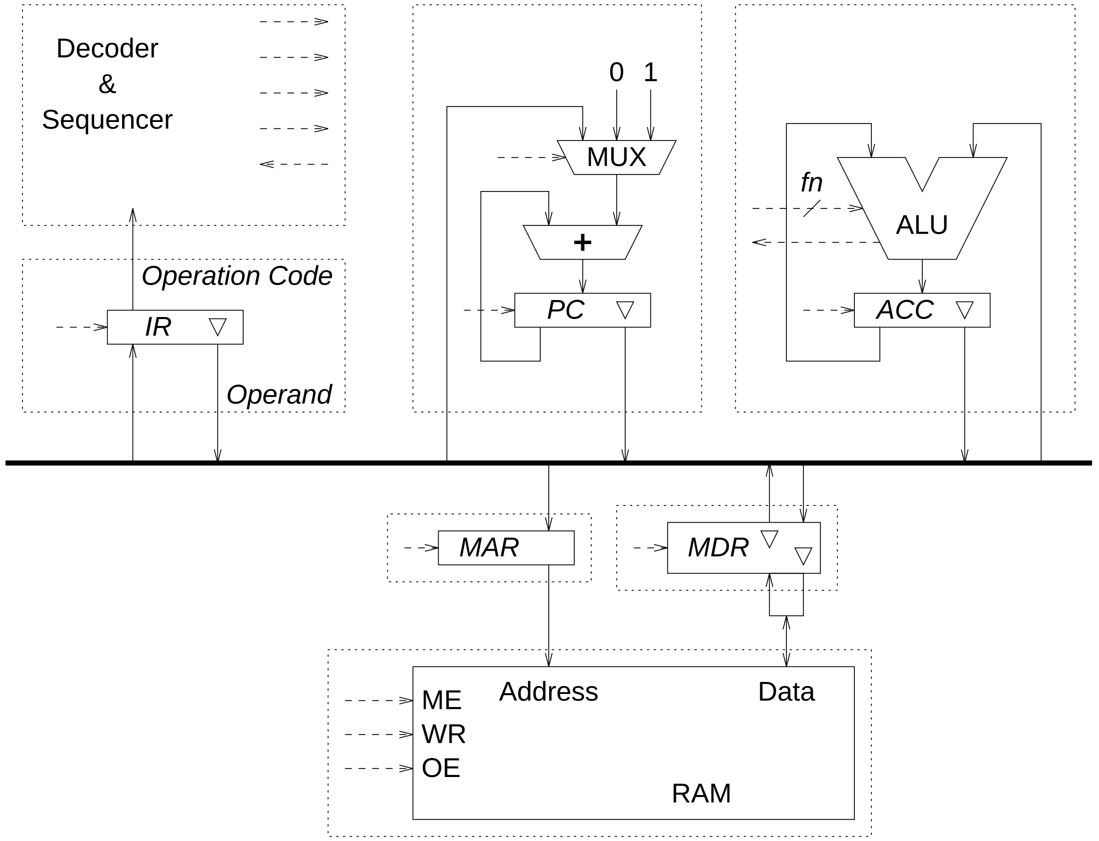
    - - Distinct Functionality
    - - Minimum Data Transfer

  - Simple Decoder & Sequencer

    - - Well Defined Instruction Cycles
    - - Simple Control
    - - Simple Feedback

Microprocessor

- Single Bus Architecture



Microprocessor

Addressing Modes

- Direct Addressing

  Operand field contains the address of the data for manipulation

  `ADD addr` $Acc' \leftarrow Acc + mem(addr)$

The use of a single bus architecture opens up the possibility of other addressing modes.

Microprocessor

Addressing Modes

- Immediate Addressing

  Operand field contains the data for manipulation

  `ADD #num` $Acc' \leftarrow Acc + num$

- Indirect Addressing

  Operand field contains the address of a pointer to the data for manipulation

  `ADD (point_addr)` $Acc' \leftarrow Acc + mem(mem(point\_addr))$

Instruction Cycle

- Instruction Fetch

  1 $MAR' \leftarrow PC$ $\qquad PC' \leftarrow PC + 1$
  2 $MDR' \leftarrow mem(MAR)$
  3 $IR' \leftarrow MDR$

  – Bus Allocation

  - - During each clock cycle only one source may drive the bus.
  - - The bus is driven by PC in cycle 1 and by MDR in cycle 3.
  - - Cycle 2 is a memory access cycle; here the bus is unused.

- Instruction Execute

  The number of cycles and the action to be taken in each cycle are dependent on the type of instruction and the addressing mode.

Instruction Cycle

- ALU instructions

Inherent	`COM`	$ACC' \leftarrow$ ~ACC
Immediate	`AND #n`	$ACC' \leftarrow ACC$ & OP
Direct	`XOR m`	$MAR' \leftarrow OP$
		$MDR' \leftarrow mem(MAR)$
		$ACC' \leftarrow ACC$ ^ MDR
Indirect	`OR (p)`	$MAR' \leftarrow OP$
		$MDR' \leftarrow mem(MAR)$
		$MAR' \leftarrow MDR$
		$MDR' \leftarrow mem(MAR)$
		$ACC' \leftarrow ACC \mid MDR$

Instruction Cycle

- Store Instructions

Direct	`STA m`	$MAR' \leftarrow OP$
		$MDR' \leftarrow ACC$
		$mem(MAR)' \leftarrow MDR$
Indirect	`STA (p)`	$MAR' \leftarrow OP$
		$MDR' \leftarrow mem(MAR)$
		$MAR' \leftarrow MDR$
		$MDR' \leftarrow ACC$
		$mem(MAR)' \leftarrow MDR$

- Branch Instructions

Unconditional	`BRA m`	$PC' \leftarrow PC + OP$
Conditional	`BCS m`	IF `(C=1)` $THEN\ PC' \leftarrow PC + OP$

Microprocessor

- Indexed Addressing

  Addresses are relative to an index register

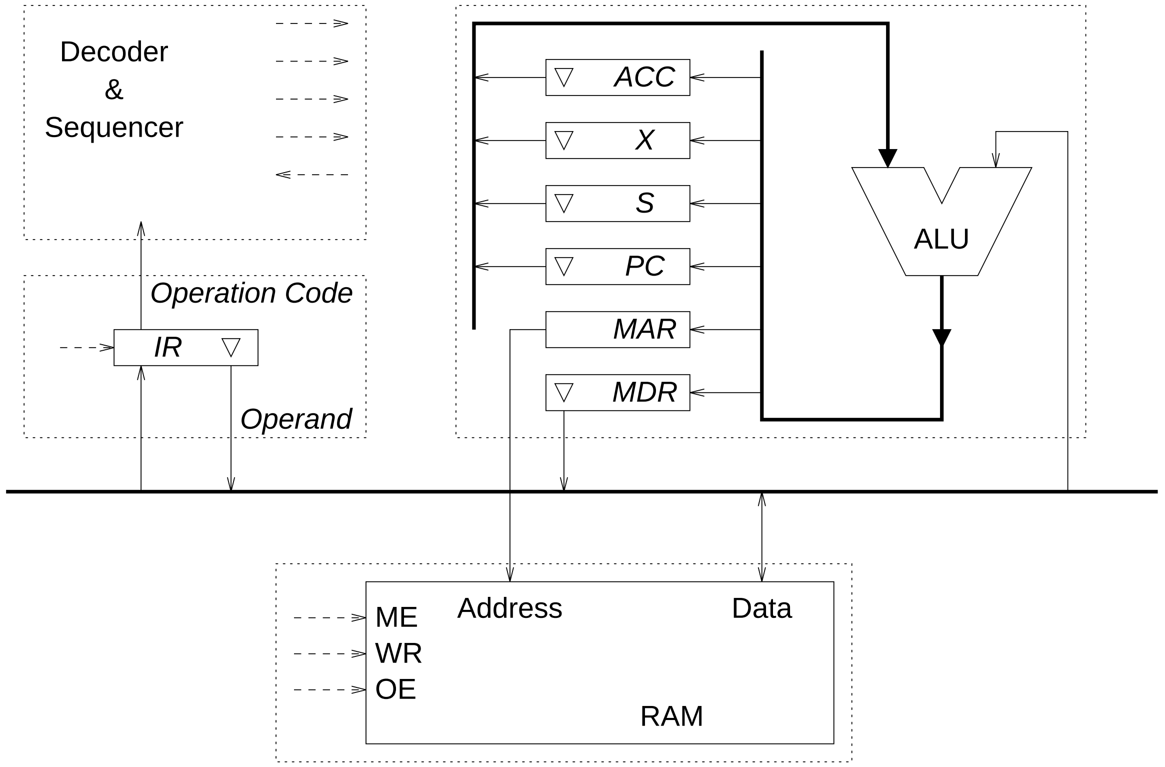
  `ADDA X,n` $Acc' \leftarrow Acc + mem(X + n)$

  – additional registers[2]
  – address calculation
    - - dedicated address generation unit?
    - - in ALU?

[2]note that we now have to specify the accumulator A for addition to avoid ambiguity.

Microprocessor

- Multiple Special Purpose Registers
- Three Busses

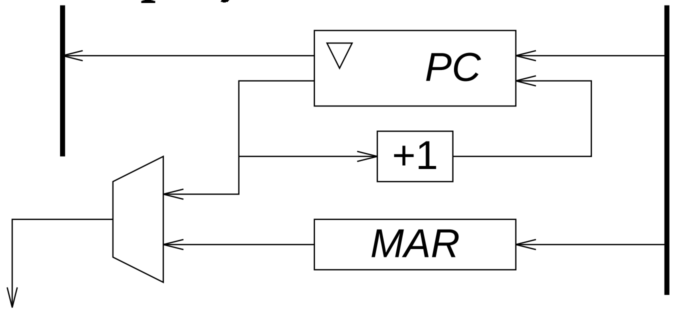


Microprocessor

ALU is shared:

- Program Counter
  $PC' \leftarrow MAR' \leftarrow PC + 1$
- Address Generation
  $MAR' \leftarrow \mathcal{F}\{X, ?\}$
- Arithmetic Logic
  $Acc' \leftarrow \mathcal{F}\{Acc, ?\}$

The results is a reduction in cost with a possible time penalty, various enhancements may be employed to reduce the number of cycles:



Microprocessor

- More addressing modes
  - Stack (Push & Pull)

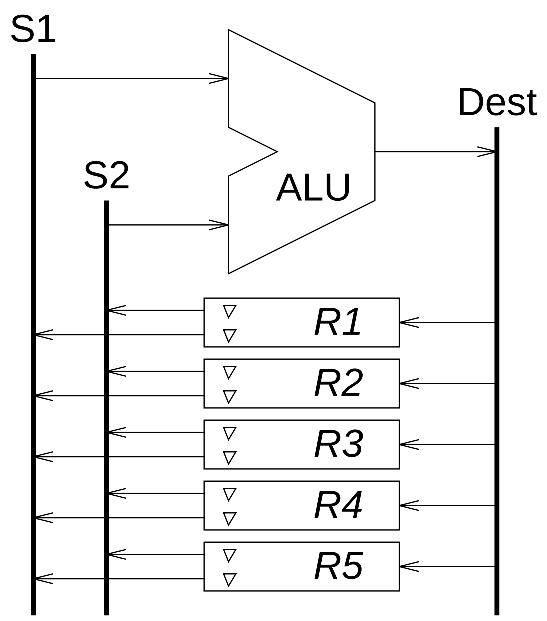
`STA -S`	$S' \leftarrow S - 1$
	$mem(S)' \leftarrow Acc$
`ADDA S+`	$Acc' \leftarrow Acc + mem(S)$
	$S' \leftarrow S + 1$

  - (Pre-)Indexed Indirect

`ADDA (X,n)`	$Acc' \leftarrow Acc + mem(mem(X + n))$

Microprocessor

- General Purpose Registers
- Register Register Architectures – Three Address Architecture

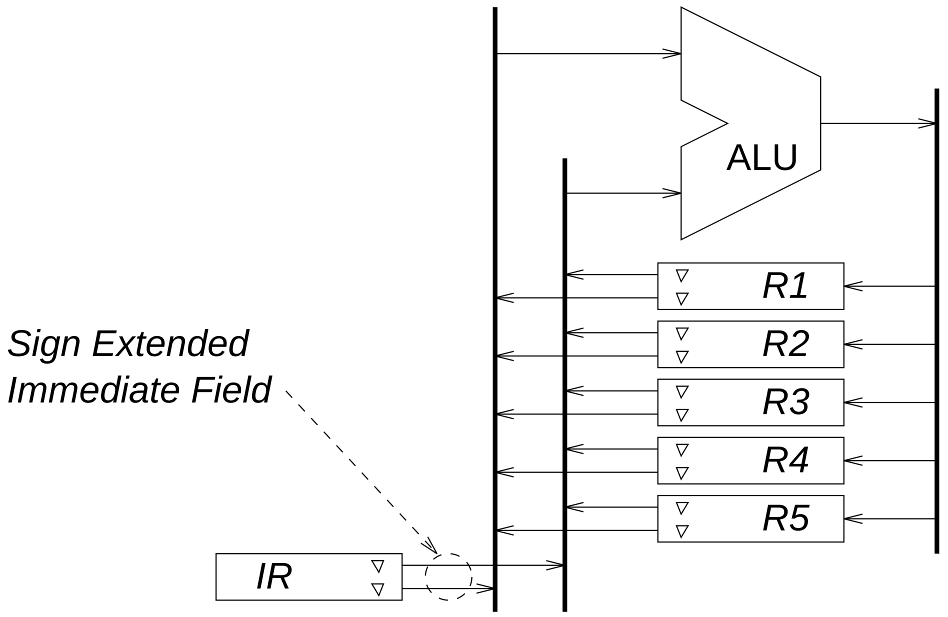


Each ALU instruction is of the form;

$$R[z]' \leftarrow \mathcal{F}\{R[x], R[y]\}.$$

Microprocessor

- Immediate Data

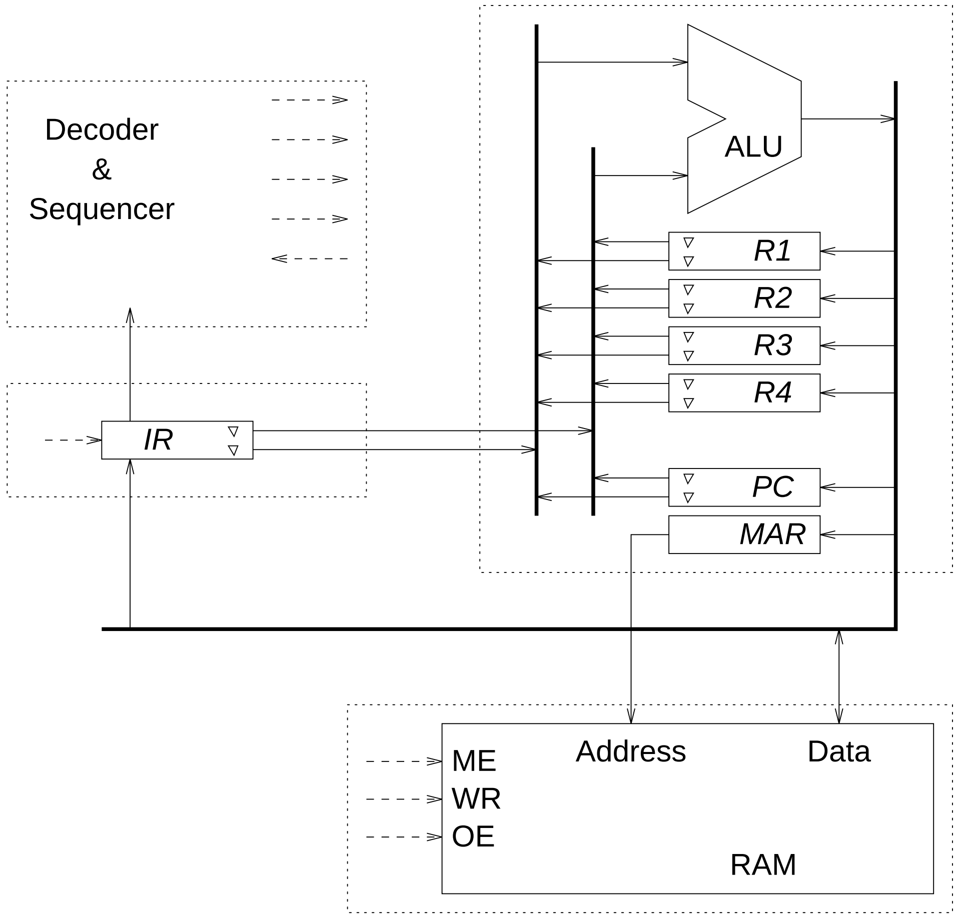


This implementation allows immediate addressing for ALU instructions;

$$R[z]' \leftarrow \mathcal{F}\{R[x], imm\}.$$

Microprocessor

- Memory Access



Microprocessor

- Memory Access

  Memory access is accomplished in two stages

  – Address generation

$$MAR' \leftarrow \mathcal{F}\{R[x], R[y]\}$$

  – Data Access

  - - Read

$$R[z]' \leftarrow mem(MAR)$$

  - - Write[3]

$$mem(MAR)' \leftarrow R[z]$$

[3]note that during memory write data is fed unmodified through the ALU.

Microprocessor

- Memory Access

Given the larger number of general purpose registers we need fewer addressing modes for load & store operations:

$$MAR' \leftarrow imm$$

or

$$MAR' \leftarrow R[x]$$

or

$$MAR' \leftarrow R[x] + imm$$

or

$$MAR' \leftarrow R[x] + R[y].$$

Microprocessor

Control transfer instructions may be PC relative or register indirect.

- PC relative

$$PC' \leftarrow PC + imm$$

- Register indirect

$$PC' \leftarrow R[x]$$

Since the latter is not limited to a sign extended offset the transfer can be to any location in memory.

Any of our general purpose registers may contain data, data address or program address.