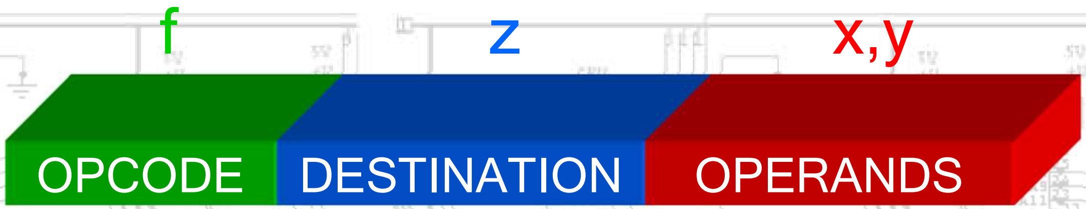
Basic Computer Architecture

- Computers consist of:
  - Datapath
  - Control unit
- It is designed to implement a particular instruction set.
- The individual instructions are the engineering equivalent of the mathematician's
  - $z=f(x,y)$



Opcode – Destination -Operands

- OPCODE
  - Selects the function
- DESTINATION
  - Is nearly always a datapath register
- OPERANDS
  - Usually come from datapath register

Instruction Format Examples

Register

31 … 15	14 … 10	9 … 5	4 … 0
Opcode	Destination Register (DR)	Source Register (SA)	Source Register (SB)

Immediate

31 … 15	14 … 10	9 … 5	4 … 0
Opcode	Destination Register (DR)	Source Register (SA)	Operand (OP)

Jump and Branch

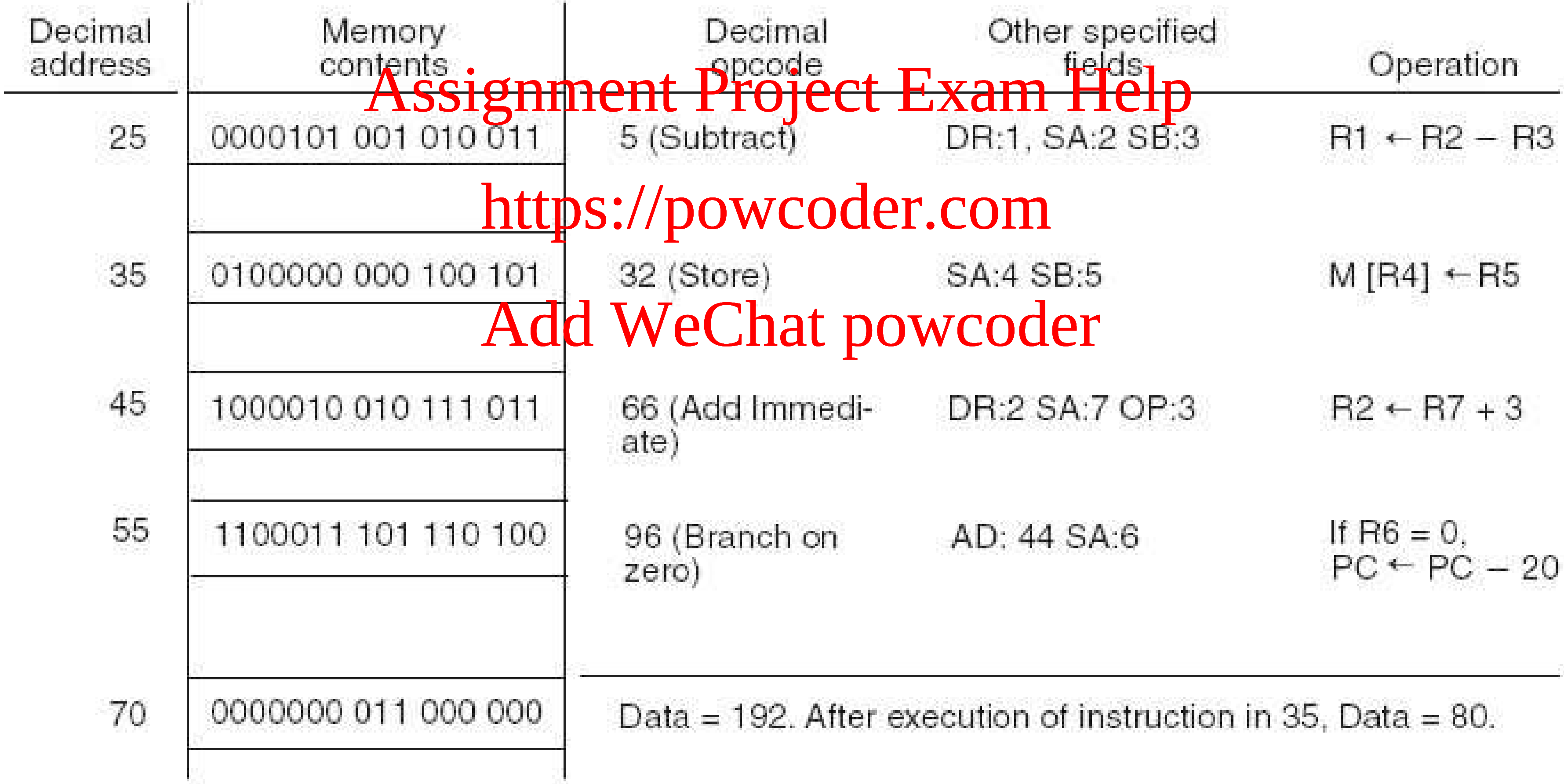
31 … 15	14 … 10	9 … 5	4 … 0
Opcode	Address (AD) (Left)	Source Register (SA)	Address (AD) (Right)

Instruction Formats

- Where DR, SA ^ SB point to processor registers in the datapath
- But Operand is itself an immediate operand

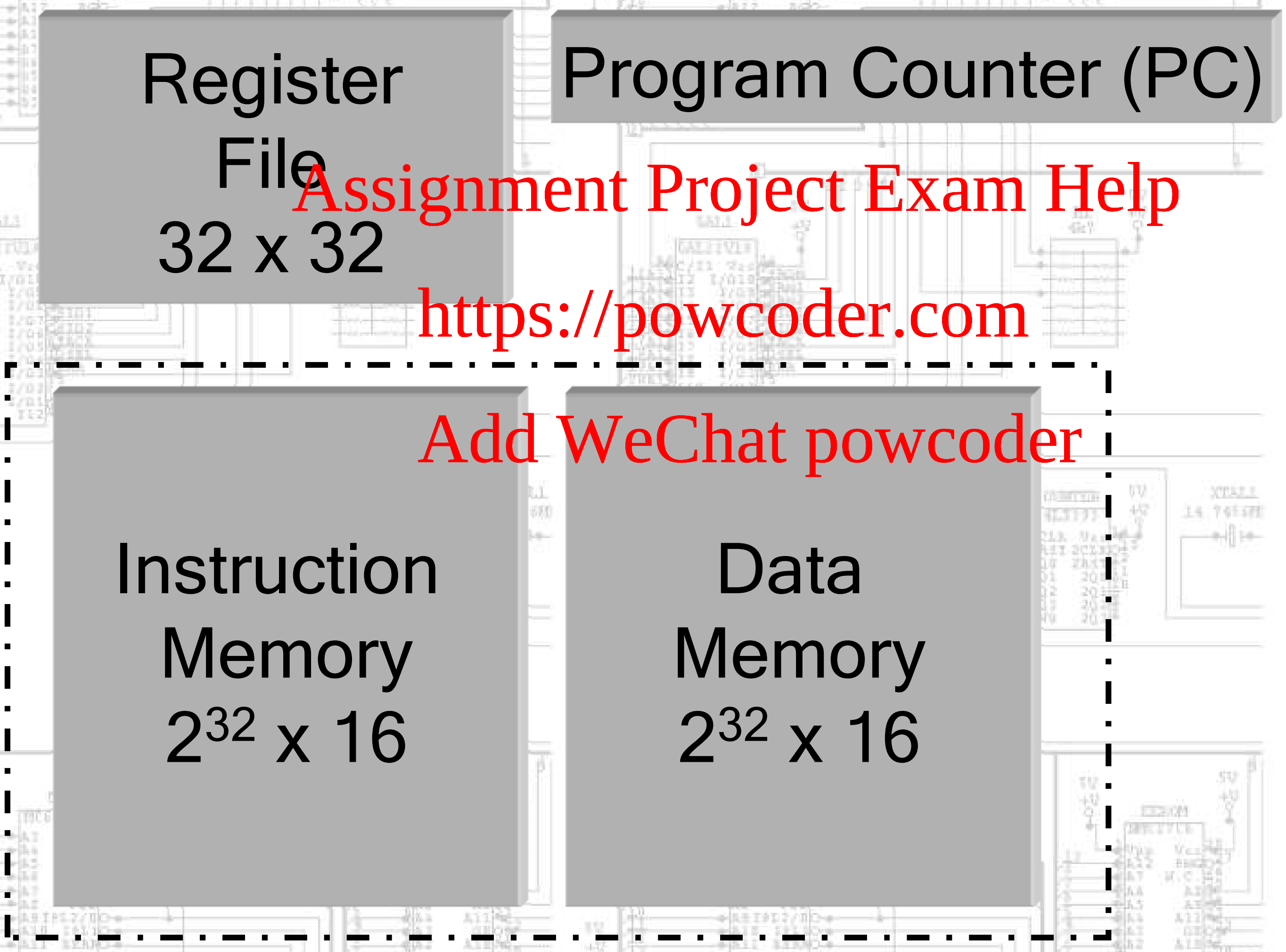
Data and Instructions in Memory

Example for a 16 Bit processor, with 7, 3, 3, 3 bits



Decimal address	Memory contents	Decimal opcode	Other specified fields	Operation
25	0000101 001 010 011	5 (Subtract)	DR:1, SA:2 SB:3	R1 ← R2 – R3
35	0100000 000 100 101	32 (Store)	SA:4 SB:5	M [R4] ← R5
45	1000010 010 111 011	66 (Add Immediate)	DR:2 SA:7 OP:3	R2 ← R7 + 3
55	1100011 101 110 100	96 (Branch on zero)	AD: 44 SA:6	If R6 = 0, PC ← PC – 20
70	0000000 011 000 000	Data = 192. After execution of instruction in 35, Data = 80.		

User View of Storage



Register File 32 x 32

Program Counter (PC)

Instruction Memory 2^{32} x 16

Data Memory 2^{32} x 16

Memory Module [entity]

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
entity memory is -- use unsigned for memory address
   Port ( address : in unsigned std_logic_vector(31 downto 0);
          write_data : in std_logic_vector(31 downto 0);
          MemWrite, MemRead : in std_logic;
          read_data : out std_logic_vector(31 downto 0));
end memory;
```

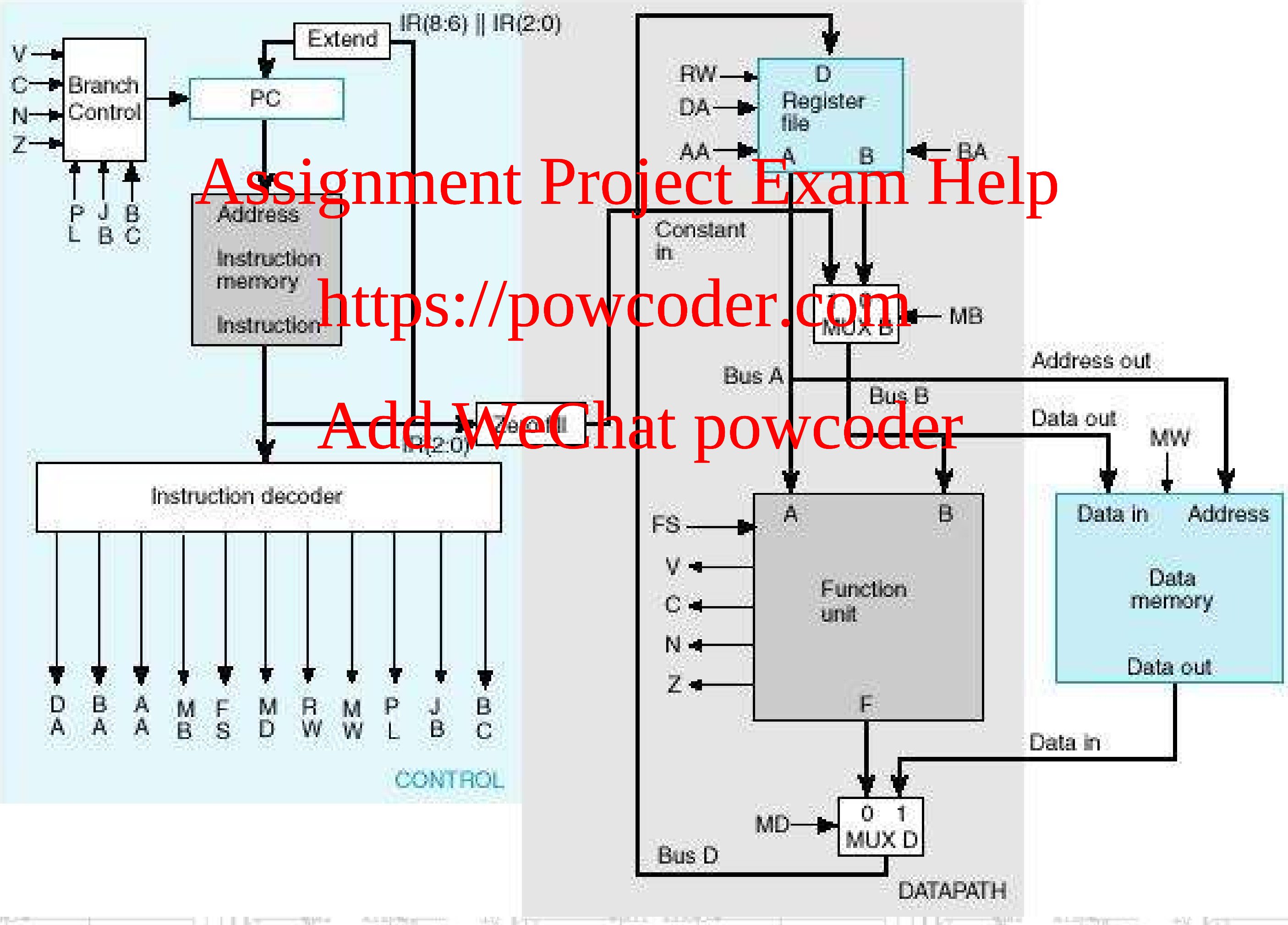
Memory Module [architecture]

```
architecture Behavioral of memory is
-- we will use the least significant 9 bit of the address - array(0 to 512)
type mem_array is array(0 to 7) of std_logic_vector(31 downto 0);
-- define type, for memory arrays
begin
mem_process: process (address, write_data)
-- initialize data memory, X denotes hexadecimal number
variable data_mem : mem_array := (
X"00000000", X"00000000", X"00000000",X"00000000",
X"00000000", X"00000000", X"00000000", X"00000000");
variable addr:integer
begin  -- the following type conversion function is in std_logic_arith
addr:=conv_integer(address(2 downto 0));
if MemWrite ='1' then
data_mem(addr):= write_data;
elsif MemRead='1' then
read_data <= data_mem(addr) after 10 ns;
end if;
end process;
end Behavioral;
```

A Single-cycle Hardwired Control Unit

- We briefly consider a system with the simplest possible control unit.
- The control unit:
  - Maps each OPCODE to a single datapath operation.
- Instructions are fetched from an instruction memory
- This is what all present systems with separate instruction and data code do.

Single-Cycle Computer



Branch

- Instruction Bit13=1
  - Jump occurs
- Instruction Bit13=0
  - Conditional Branch occurs
  - Bit11, Bit10 and Bit9
    - Select the status bit

BC	Status Bit
000	C
001	N
010	V
011	Z
100	$\overline{C}$
101	$\overline{N}$
110	$\overline{V}$
111	$\overline{Z}$

Jump and Branch

- PL=1
  - Jump or Branch, loading the PC

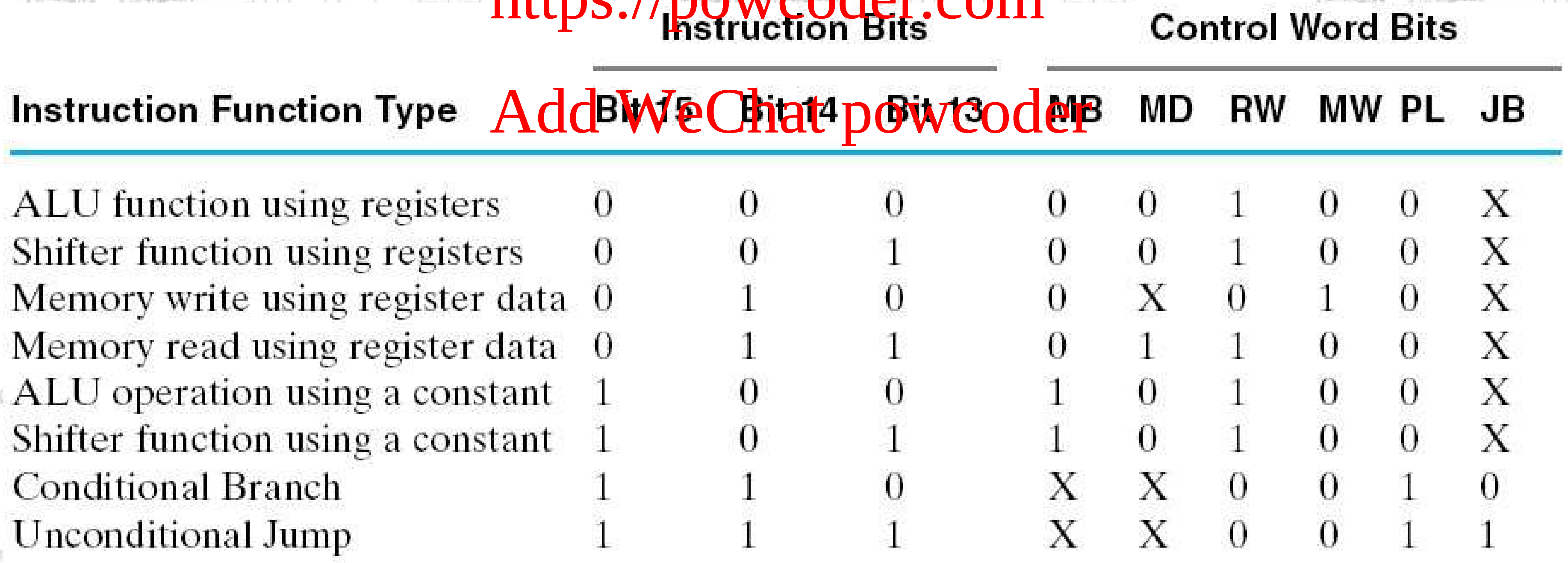
- PL=0

  - PC is incremented

- PL=1 ∧ JB=0
  - Jump
- PL=1 ∧ JB=1
  - Conditional branch

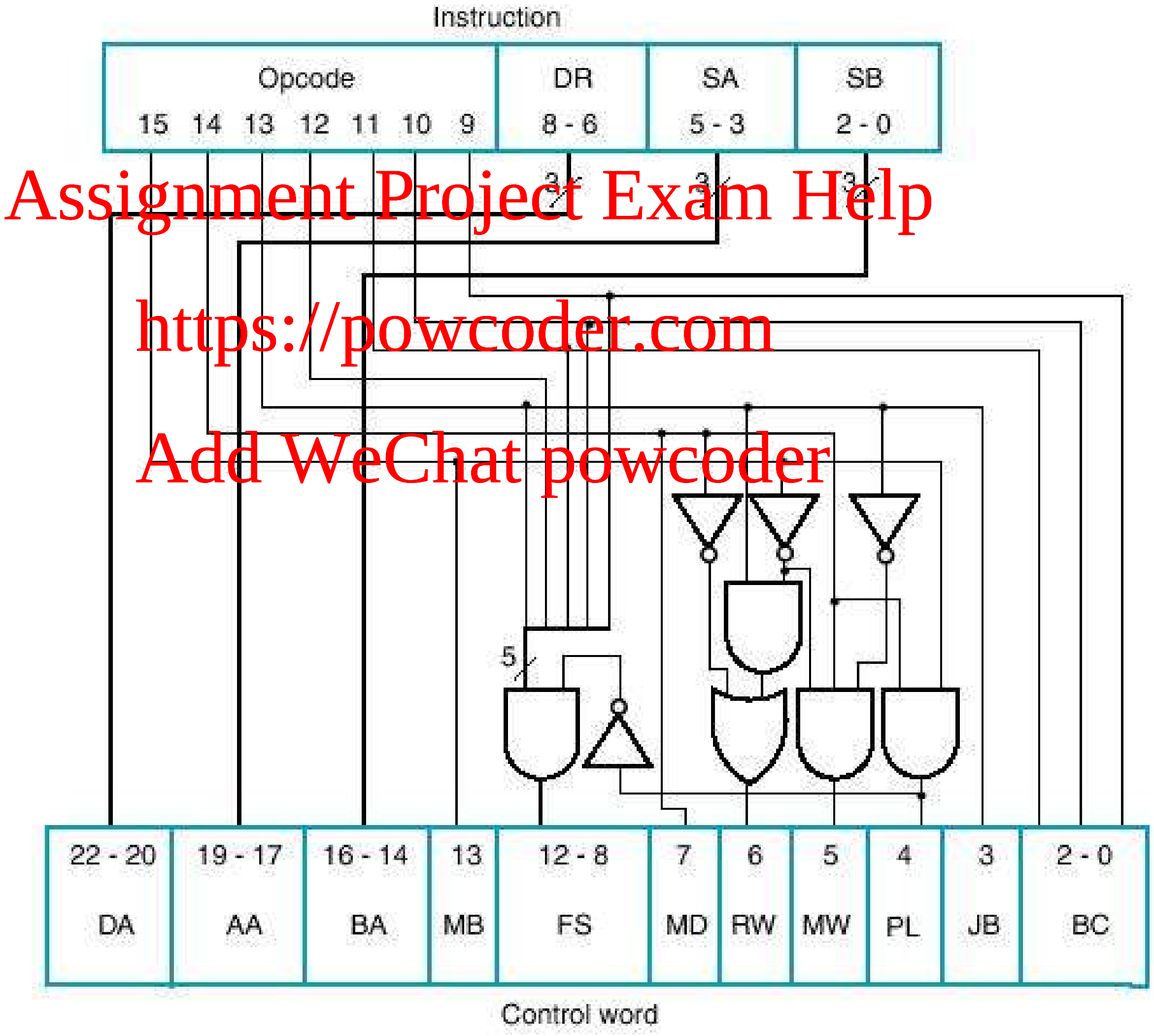
Truth Table BIT15 – BIT13

- The following operations classification helps with the implementation of the instruction decoder



	Instruction Bits			Control Word Bits					
Instruction Function Type	Bit 15	Bit 14	Bit 13	MB	MD	RW	MW	PL	JB
ALU function using registers	0	0	0	0	0	1	0	0	X
Shifter function using registers	0	0	1	0	0	1	0	0	X
Memory write using register data	0	1	0	0	X	0	1	0	X
Memory read using register data	0	1	1	0	1	1	0	0	X
ALU operation using a constant	1	0	0	1	0	1	0	0	X
Shifter function using a constant	1	0	1	1	0	1	0	0	X
Conditional Branch	1	1	0	X	X	0	0	1	0
Unconditional Jump	1	1	1	X	X	0	0	1	1

Instruction Decoder



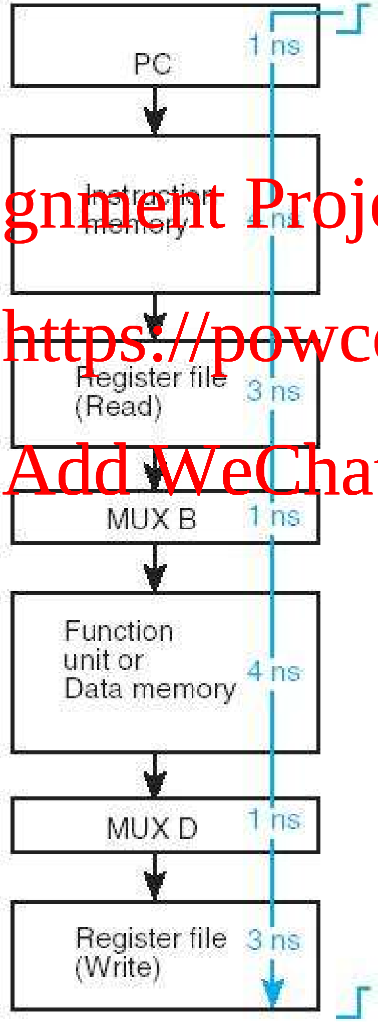
Single-Cycle Computer Instruction Example

Operation code	Symbolic name	Format	Description	Function	MB	MD	RW	MW	PL	JB
1000010	ADI	Immediate	Add immediate operand	$R[\mathrm{DR}] \leftarrow R[\mathrm{SA}] + \mathrm{zf}\ I(2{:}0)$	1	0	1	0	0	0
0110000	LD	Register	Load memory content into register	$R[\mathrm{DR}] \leftarrow M[R[\mathrm{SA}]]$	0	1	1	0	0	1
0100000	ST	Register	Store register content in memory	$M[R[\mathrm{SA}]] \leftarrow R[\mathrm{SB}]$	0	1	0	1	0	0
0011000	SL	Register	Shift left	$R[\mathrm{DR}] \leftarrow \mathrm{sl}R[\mathrm{SB}]$	0	0	1	0	0	1
0001110	NOT	Register	Complement register	$R[\mathrm{DR}] \leftarrow \overline{R[\mathrm{SA}]}$	0	0	1	0	0	0
1100000	BRZ	Jump/Branch	If $R[\mathrm{SA}] = 0$, branch to PC + se AD	If R[SA] = $0, PC \leftarrow PC + \mathrm{se}AD$, If R[SA] $\neq 0, PC \leftarrow PC + 1$	1	0	0	0	1	0

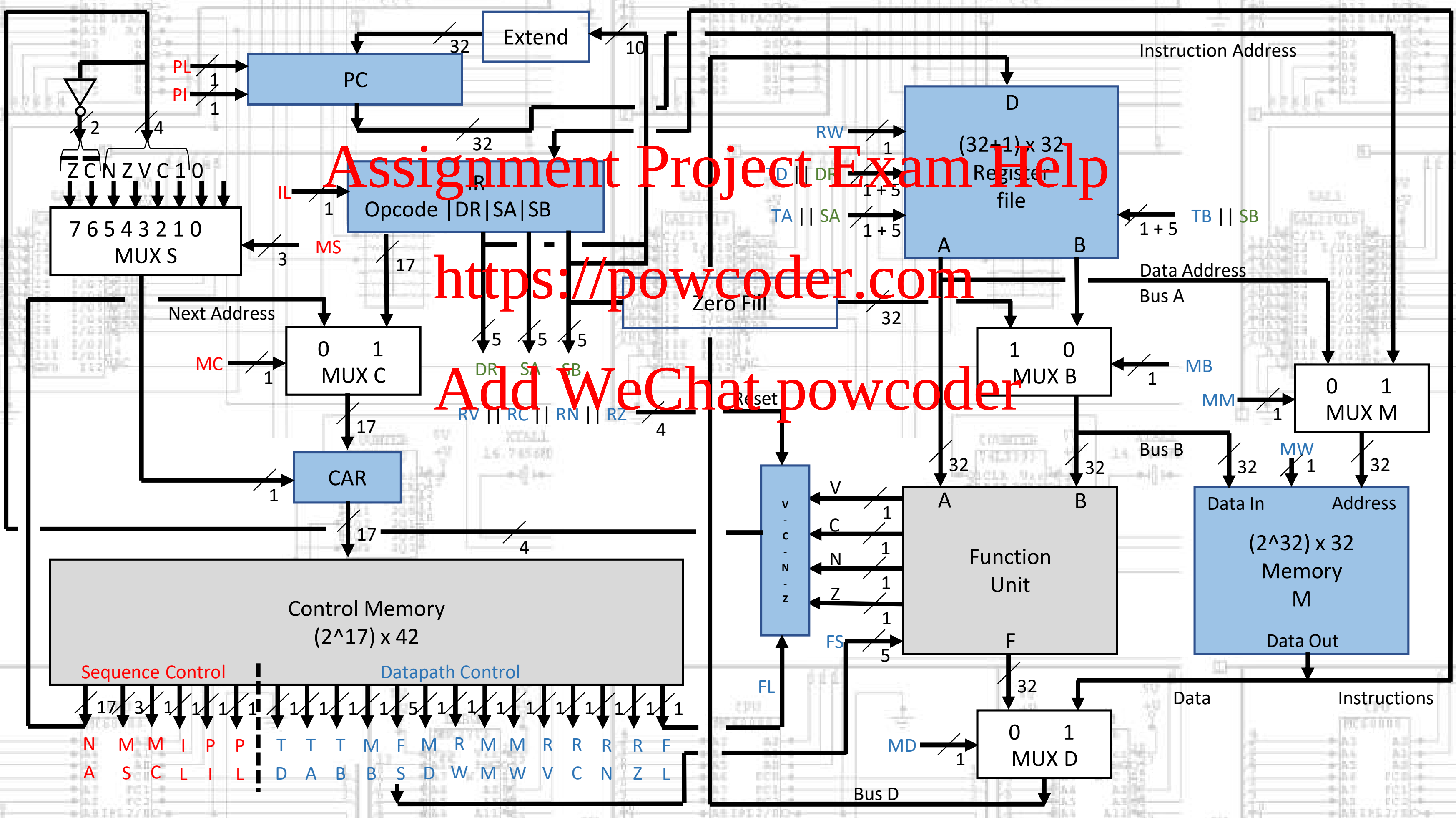
Single-cycle Problem

- A single-cycle control unit cannot implement:
  - more complex addressing modes
  - Composite functions
    - E.g. Multiplication
- A single-cycle control unit has long worst case delay path.
  - Slow clock.

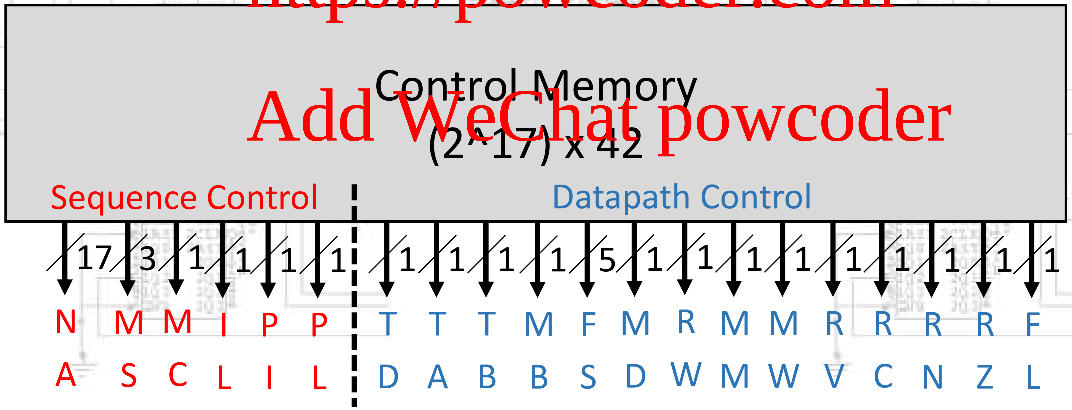
Worst Case Delay



Multiple-Cycle Microprogrammed Computer



Microinstruction Format



Control Word Information for Datapath

Example, our project is different!

<table>
<tr><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">NA</td><td colspan="3">MS</td><td>MC</td><td>IL</td><td>PI</td><td>PL</td><td>TD</td><td>TA</td><td>TB</td><td>MB</td><td colspan="5">FS</td><td>MD</td><td>RW</td><td>MM</td><td>MW</td></tr>
</table>

TD	TA	TB	MB		FS		MD	RW	MM	MW	
Select	Select	Select	Select	Code	Function	Code	Select	Function	Select	Function	Code
$R[DR]$	$R[SA]$	$R[SB]$	Register	0	$F = A$	00000	FnUt	No write (NW)	Address	No write (NW)	0
$R8$	$R8$	$R8$	Constant	1	$F = A + 1$	00001	Data In	Write (WR)	PC	Write (WR)	1
					$F = A + B$	00010					
					$F = A + B + 1$	00011					
					$F = A + \overline{B}$	00100					
					$F = A + \overline{B} + 1$	00101					
					$F = A - 1$	00110					
					$F = A$	00111					
					$F = A \wedge B$	01000					
					$F = A \vee B$	01010					
					$F = A \oplus B$	01100					
					$F = \overline{A}$	01110					
					$F = B$	10000					
					$F = \text{sr } B$	10100					
					$F = \text{sl } B$	11000					