ECE 52 Protocomputer
Controller
MBR

**BUS[0..15]**
- \(0\)
- 16-bit
- 2:1 MUX
- \(1\)

**MEM_OUT[0..15]**

**MBR_SELECT**

**MBR_OUT_EN**

**MBR_IN_EN**

*This register has two types of output: (A) 3-state and (B) direct logic*
SP (or PC*)

RESET*

BUS[4..15]

12-bit Counter

U/D LOAD EN

3-state Output Buffers EN

SP_U/D' SP_IN_EN SP_INC_EN SP_OUT_EN

BUS[4..15] STATUS
IR

BUS[0..15]

16-bit Register*

IR_OUT_EN

IR_IN_EN

OUT_EN

IN_EN

3-state Output Buffers

BUS[4..15]

DECODE[0..15]

EN

IR_OUT_EN

* This register has two types of output: (A) 3-state and (B) direct logic
* This register has two types of output: (A) 3-state and (B) direct logic
STATUS (k bits)

* This register has two types of output: (A) 3-state and (B) direct logic


**Fetch Cycle**

1. \( \text{MAR} \leftarrow \text{PC} \) ; move address of next instr to MAR
2. \( \text{MBR} \leftarrow \text{MEM}[	ext{MAR}] \) ; i.e. READ!
3. \( \text{IR} \leftarrow \text{MBR} \) ; move instruction to IR
4. \( \text{PC} \leftarrow \text{PC} + 1; \ S_1\_\text{SET}; \ S_0\_\text{RESET} \) ; to avoid infinite loop

\[
\begin{align*}
F & \quad \text{PC\_OUT\_EN0} \\
\text{PHI0} & \quad \text{MAR\_IN\_EN0} \\
\text{PC\_OUT\_EN} & \quad \text{MAR\_IN\_EN}
\end{align*}
\]
Fetch Cycle

1. MAR ← PC ; move address of next instr to MAR
2. MBR ← MEM[MAR] ; i.e. READ!
3. IR ← MBR ; move instruction to IR
4. PC ← PC + 1; S1_SET; S0_RESET ; to avoid infinite loop
Fetch Cycle

1. MAR ← PC ; move address of next instr to MAR
2. MBR ← MEM[MAR] ; i.e. READ!
3. IR ← MBR ; move instruction to IR
4. PC ← PC + 1; S1_SET; S0_RESET ; to avoid infinite loop
Fetch Cycle

1. MAR ← PC ; move address of next instr to MAR
2. MBR ← MEM[MAR] ; i.e. READ!
3. IR ← MBR ; move instruction to IR
4. PC ← PC + 1; S1_SET; S0_RESET ; to avoid infinite loop

---

```
F
PHI3
```

```
PC_INC_EN0
...  S1_SET0
...  ...  MBR_OUT_EN
S1_SET
...  S0_RESET0
S0_RESET
...  ...
```
Execute Cycle: READ

1. MAR ← IR[4..15] ; move only 12 LSB of IR to MAR
2. MBR ← MEM[MAR] ; read number from memory
3. AC ← MBR ; move number where we need it
4. S1_RESET; S0_SET
Execute Cycle: READ

1. \( \text{MAR} \leftarrow \text{IR}[4..15] \) ; move only 12 LSB of IR to MAR
2. \( \text{MBR} \leftarrow \text{MEM}[\text{MAR}] \) ; read number from memory
3. \( \text{AC} \leftarrow \text{MBR} \) ; move number where we need it
4. \( \text{S1_RESET}; \text{S0_SET} \)

\[ \begin{align*}
\text{E} & \quad \text{MBR\_SELECTx} \\
\text{PHI1} & \quad \ldots \\
\text{READ} & \quad \text{MBR\_SELECT} \\
\text{MBR\_IN\_ENx} & \quad \ldots \\
\text{MBR\_IN\_EN} & 
\end{align*} \]
Execute Cycle: READ

1. MAR ← IR[4..15]  ; move only 12 LSB of IR to MAR
2. MBR ← MEM[MAR]  ; read number from memory
3. AC ← MBR        ; move number where we need it
4. S1_RESET; S0_SET
Execute Cycle: READ

1. MAR ← IR[4..15] ; move only 12 LSB of IR to MAR
2. MBR ← MEM[MAR] ; read number from memory
3. AC ← MBR ; move number where we need it
4. S1_RESET; S0_SET