# 4.Instruction Execution (Multi-Cycle Implementation)

# Contents

# 4.1 Limitation of Single-Cycle Implementation

- Different types of instructions typically need different number of computation steps.

| Stage | `rrmov rA,rB = 2\|0\|rA\|rB` | |
|---|---|---|
| Fetch | *icode*, *ifun* | *icode*:*ifun* $\leftarrow M_1[PC]$ |
| | *rA*, *rB* | *rA*:*rB* $\leftarrow M_1[PC+1]$ |
| | *valC* | |
| | *valP* | $valP \leftarrow PC+2$ |
| Decode | *valA*, *srcA* | $valA \leftarrow R[rA]$ |
| | *valB*, *srcB* | |
| Execute | *valE* | $valE \leftarrow valA+0$ |
| | *cond* | |
| Memory | *valM* | |
| Write back | *dstE* | $R[rB] \leftarrow valE$ |
| | *dstM* | |
| PC update | *PC* | $PC \leftarrow valP$ |

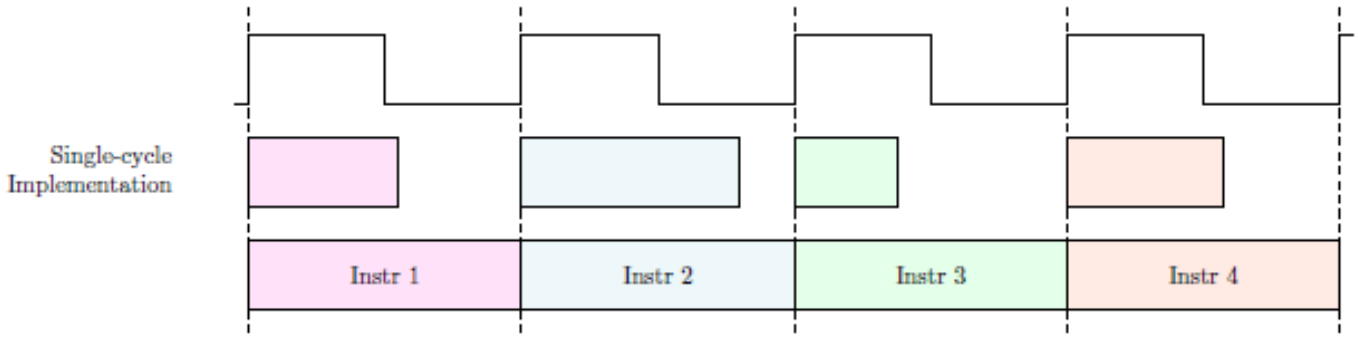
| Stage | `jXX = 7\|X\|D` | |
|---|---|---|
| Fetch | *icode*, *ifun* | *icode*:*ifun* $\leftarrow M_1[PC]$ |
| | *rA*, *rB* | |
| | *valC* | $valC \leftarrow M_4[PC+1]$ |
| | *valP* | $valP \leftarrow PC+9$ |
| Decode | *valA*, *srcA* | |
| | *valB*, *srcB* | |
| Execute | *valE* | |
| | *cond* | $branch \leftarrow Check(cond, ifun)$ |
| Memory | *valM* | |
| Write back | *dstE* | |
| | *dstM* | |
| PC update | *PC* | $PC \leftarrow valC$ if$(branch = 1)$ else $valP$ |

# 4.1 Limitation of Single-Cycle Implementation

- The single – cycle implementation is designed to use only one cycle for every type of instructions.

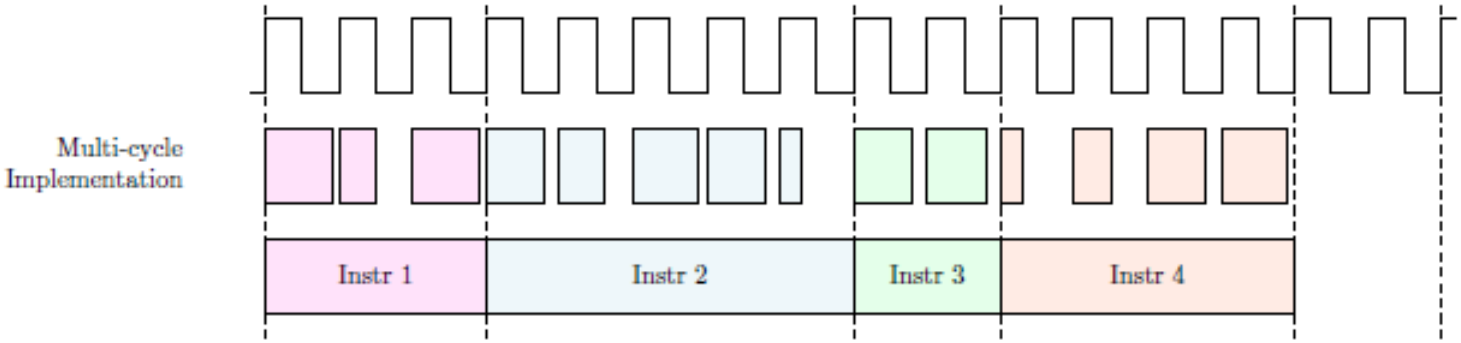
- The clock period needs to cover all the computation steps, although many instructions need only subset of the steps.

- The performance of the processor can be improved if we can shorten the execution period for some types of instructions.



# 4.2 Multi-Cycle Implementation

- The multi – cycle implementation is designed to perform only subset of the computation steps in one cycle.
- Therefore, the clock period can be made shorter than the clock period of the single – cycle implementation.
- However, a single instruction needs multiple cycles to complete all the required steps.



# 4.2 Multi-Cycle Implementation

- In the multi – cycle implementation, all types of instructions do not need to use the same number of cycles for the execution.

- Simple instructions can be executed in a fewer number of cycles than sophisticated instructions.

- Thus, this improves the performance of the processor.

# 4.2 Multi-Cycle Implementation

**Single – cycle implementaion**

- One clock cycle is for executing one instruction.
- Every instruction type requires the same execution time.

**Multi – cycle implementation**

- One clock cycle is for executing one stage.
- Different instruction types require different number of cycles, i.e. different execution time.

# 4.2.1 Datapath

- We need to introduce a number of **temporary storages** in the datapath.

- They are necessary to store the values through the multiple cycles that an instruction is executed.

- We set these new control signals to “1” when we want to update the temporary storages in the next cycle.
- If we set them to “0”, the values of the temporary storages remain unchanged.

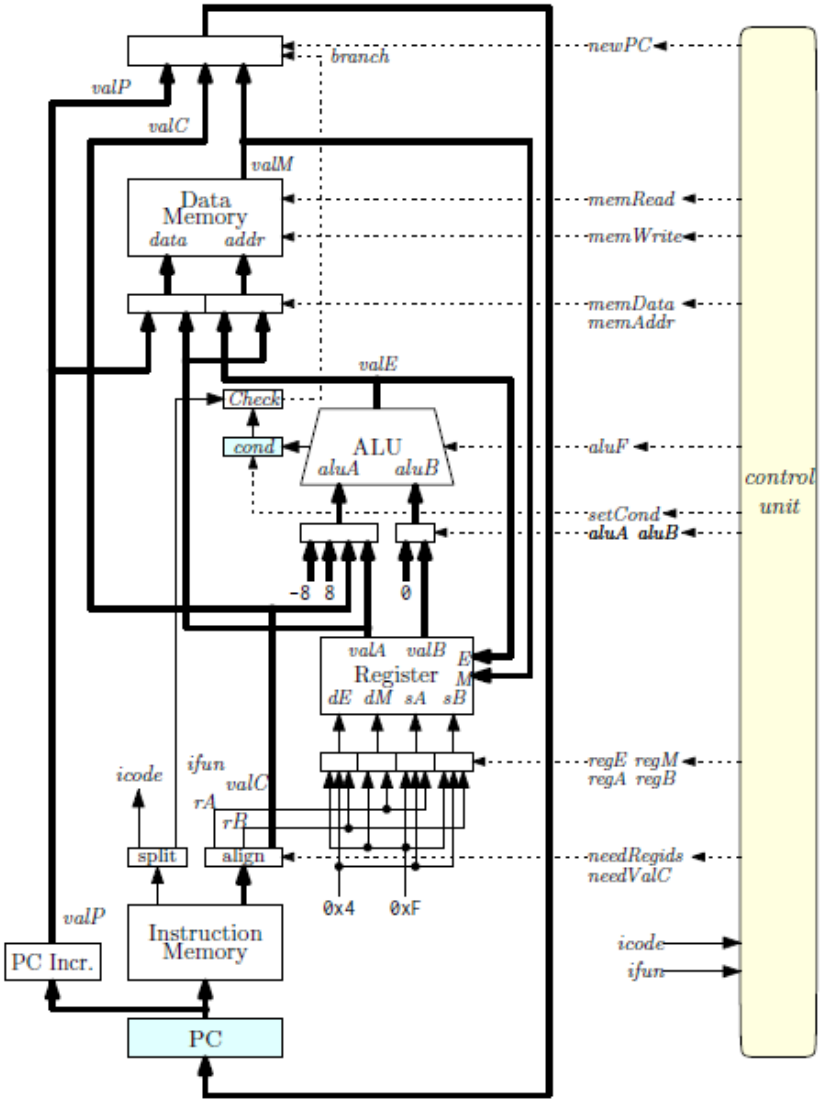
# 4.2.1 Datapath

1. Instruction Register – it is used to stored the fetched instruction.

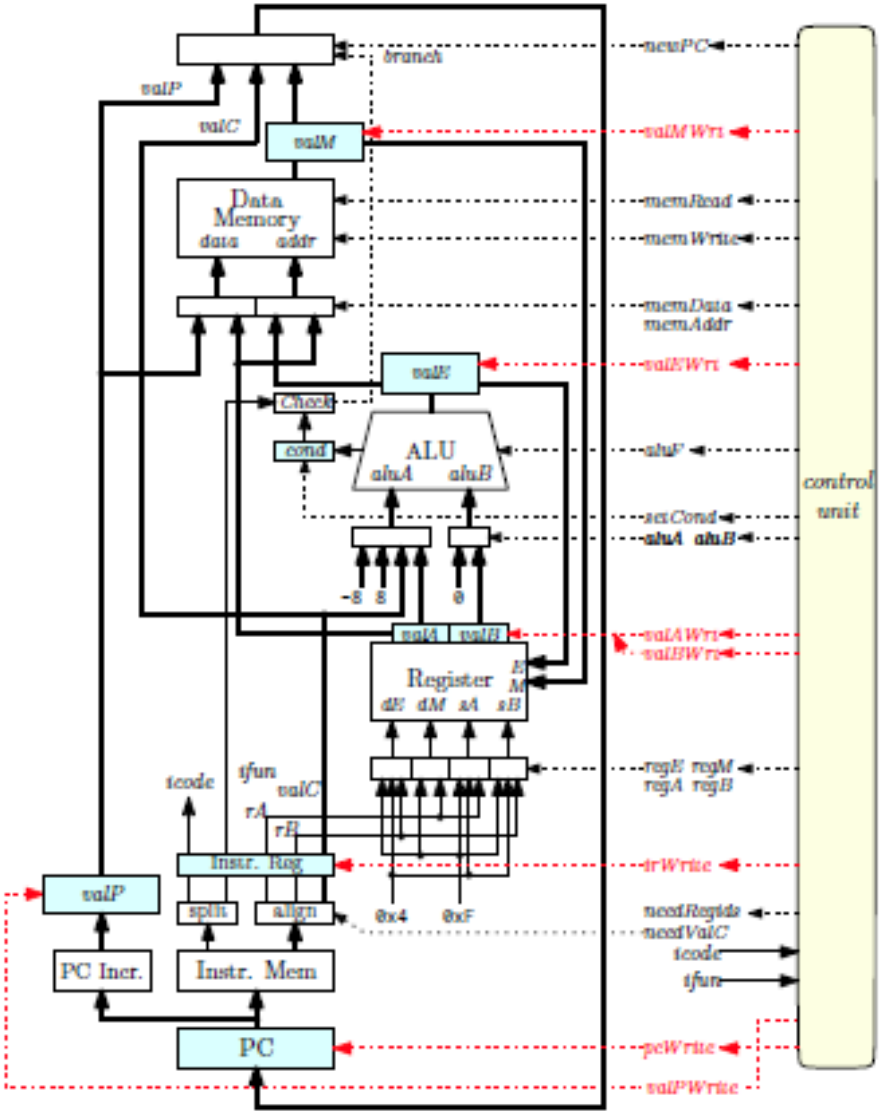
   The irWrite control signal – it can be set to 1 when the register needs updating.
2. valP and valPWrite – they are a storage and a control signal for the address of the next instruction.
3. valA and valB – they are storages from the values obtained from the register file.

   valAWrite and valBWrite – we use them to control the updating.
4. valE and valM – They are storages as well.

   valEWrite and valMWrite – we use them to control the updating.
5. pcWrite control signal – it is added to update the PC only the cycle performing PC Update.




Single – Cycle Implementation




Multi – Cycle Implementation

# Example 4.1

- Show how a nop instruction is executed.

# Example 4.2

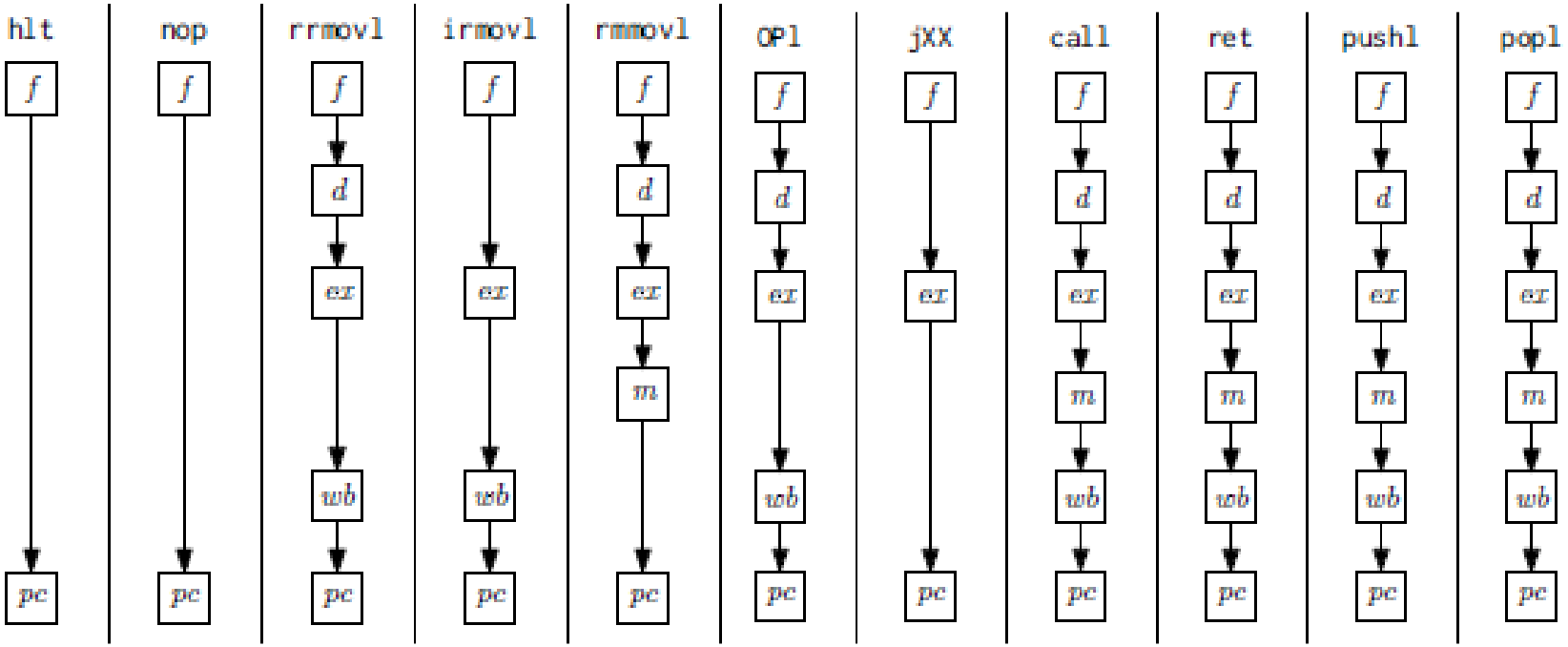
- Show how a irmov V,rB instruction is executed.

# Exercise 4.1

- Write the control signal when mrmov D(rB),rA is executed.

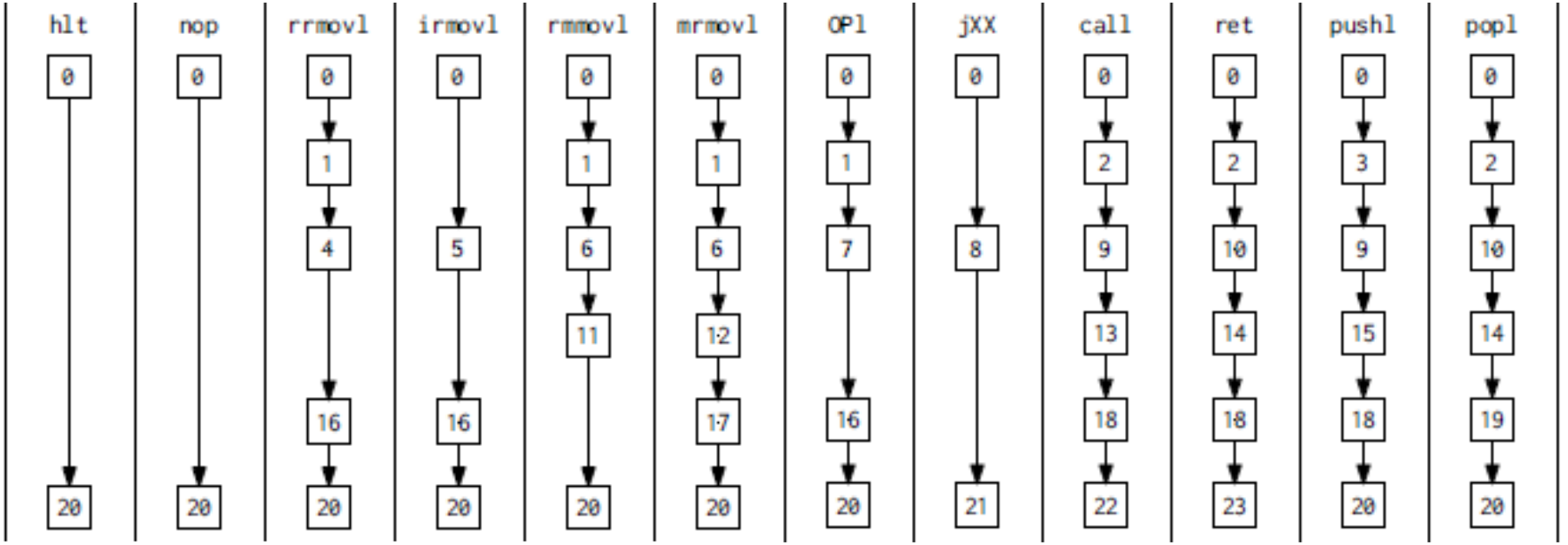
# 4.2.2 Control Unit

- To design a control unit for the multi – cycle implementation, we first need to check how each type of instruction is executed.
- The following figure summarizes the execution stages used in each type of instructions.



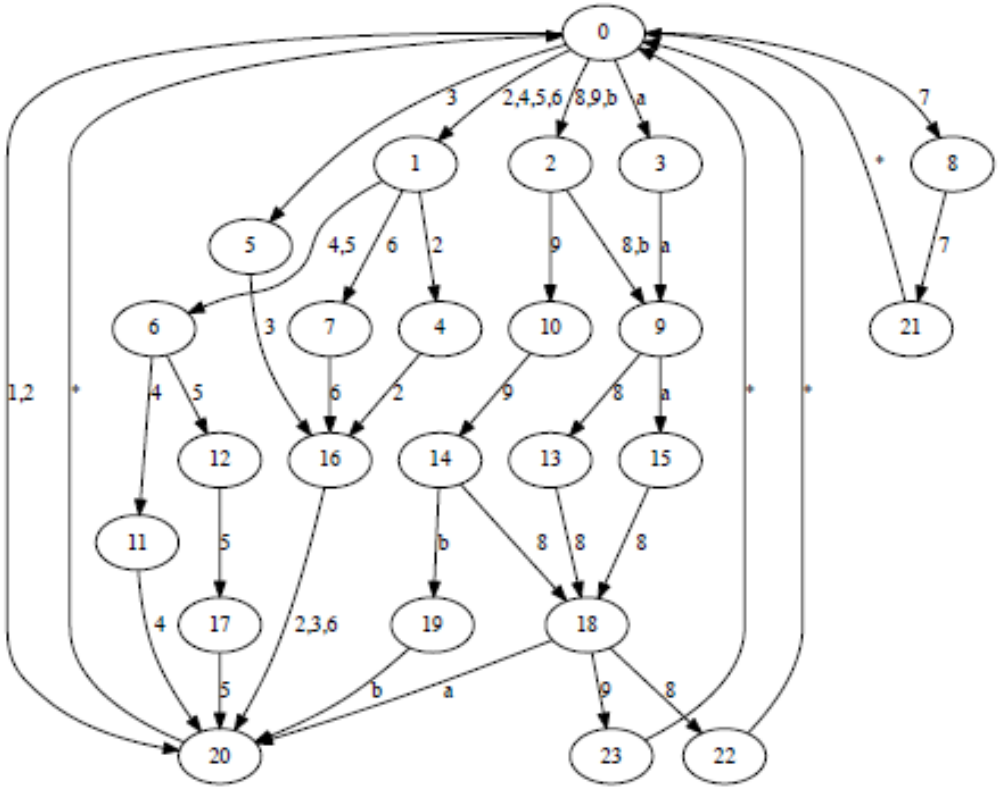
# 4.2.2 Control Unit

- Then, we assign a state number to each computation steps.
- We use the same number for the steps performing the same or similar task.



# 4.2.2 Control Unit

- The control unit can be designed as a state machine that emits the control signals based on the current state and the instruction code (icode).
- This can be later transformed to a synchronous digital circuit for the control unit.



# 4.2.3 Determining Clock Period

**Single – cycle implementation**

- The clock period (P) needs to be long enough to complete all the stages in the entire datapath.

$$P = P_f + P_d + P_e + P_m + P_{wb} + P_{pc}$$

- CPI (Clock cycle per instruction) = 1

**Multi – cycle implementation**

- The clock period needs to be long enough to complete the longest stage in the datapath.

$$P = \max \{P_f, P_d, P_e, P_m, P_{wb}, P_{pc}\}$$

# 4.2.3 Determining Clock Period

- Exercise 4.3 From the following required time period for each computation stage, what are the clock period for the single – cycle and multi – cycle implementation of Y86?

| Stage | Required Time Period |
|---|---|
| Fetch | 0.3 ns |
| Decode | 0.2 ns |
| Execute | 0.3 ns |
| Memory | 0.4 ns |
| Write back | 0.2 ns |
| PC Update | 0.1 ns |

# 4.2.3 Determining Clock Period

- Exercise 4.4 What is the execution time and CPI of the following program when the clock period is set to 0.5 ns?

```
push %rbx
push %rsi
irmov $0,%rbx
irmov $5,%rsi
add %rbx,%rsi
pop %rsi
pop %rbx
ret
```

# 4.3 Microprogramming

- A simple technique used to implement the control units is called “hardwired control implementation”.
- This technique has a limitation that changing a part in the datapath causes the circuit to be redesigned.

- Another technique is called “microprogramming”. This is a technique to implement the control units by using ROM to store the collection of control signals corresponding to the state, as well as the next state.
- Therefore, the control unit can be changed without redesigning the entire circuit.

# References


- C.Nattee, Lecture Note of CSS224 Computer Architectures, 2017.