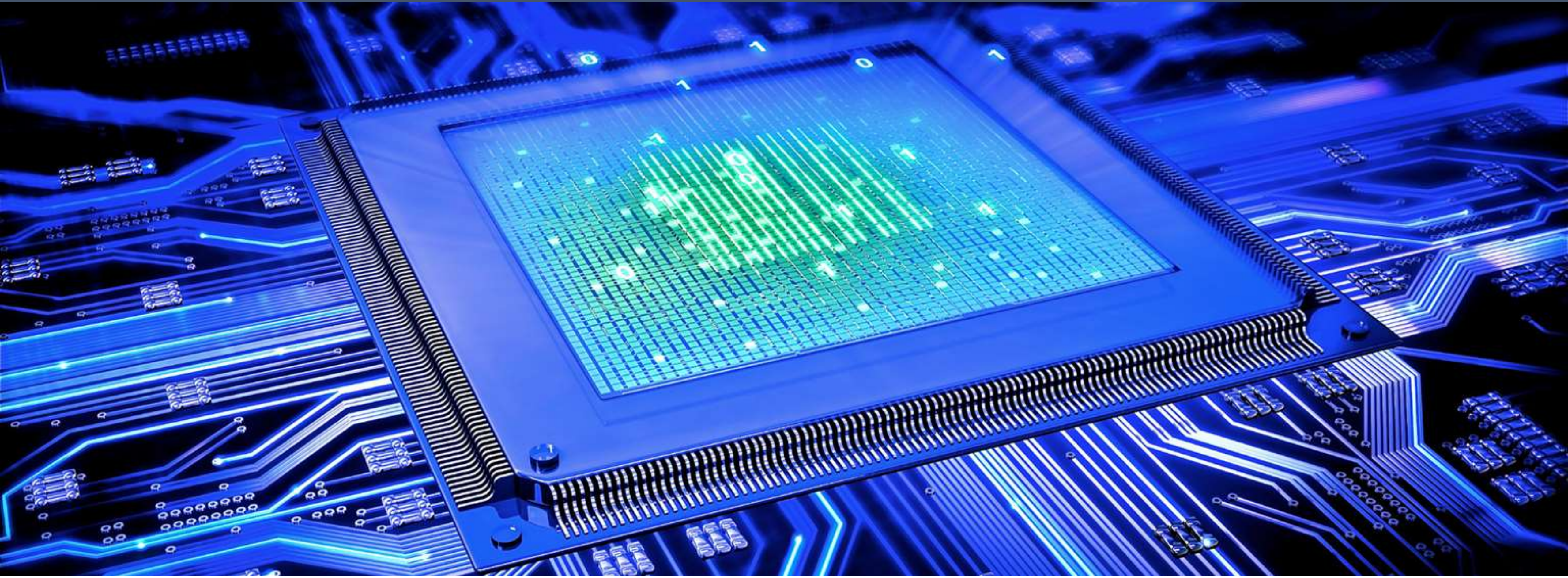
Revision



Unit 11.1B – Programming paradigms

Lesson objectives

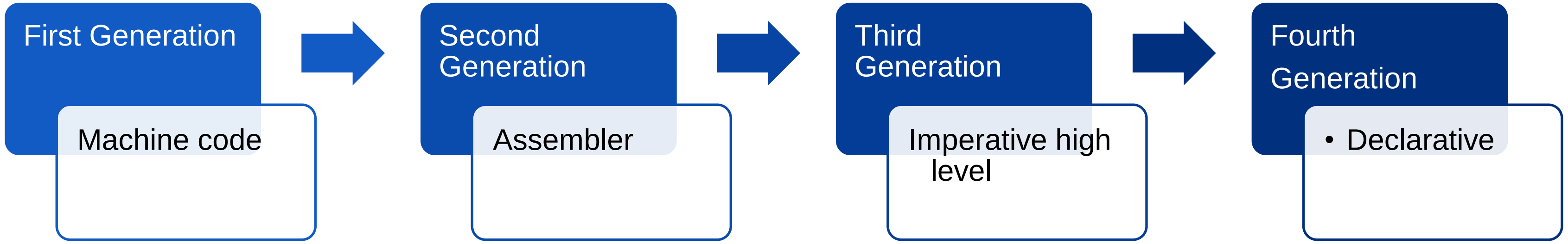
01 distinguish between generations of programming languages

02 classify programming languages into low and high-level

03 analyze the advantages and disadvantages of high-level languages

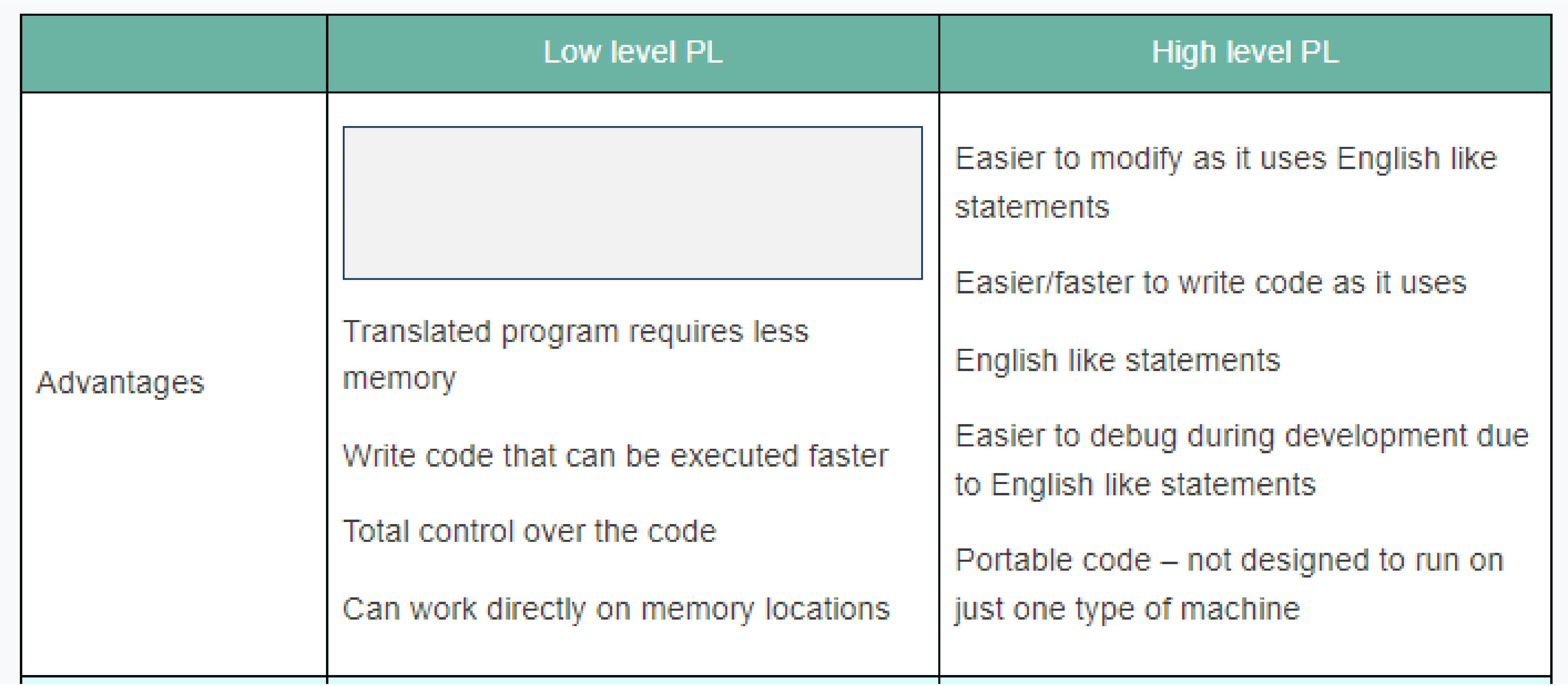
04 analyze the advantages and disadvantages of low-level languages

In the early days of computing, a computer could only be programmed using machine code. This was a difficult and tedious task even to code up the most simple algorithms. Then the assembly language was developed. It was easier to code using Assembly than machine code but it was still difficult. Later imperative high level programming languages were developed that made coding accessible to many more people as programming was now much easier. Machine code, assembler and imperative high level programming languages are referred to as first, second and third generation programming languages respectively.

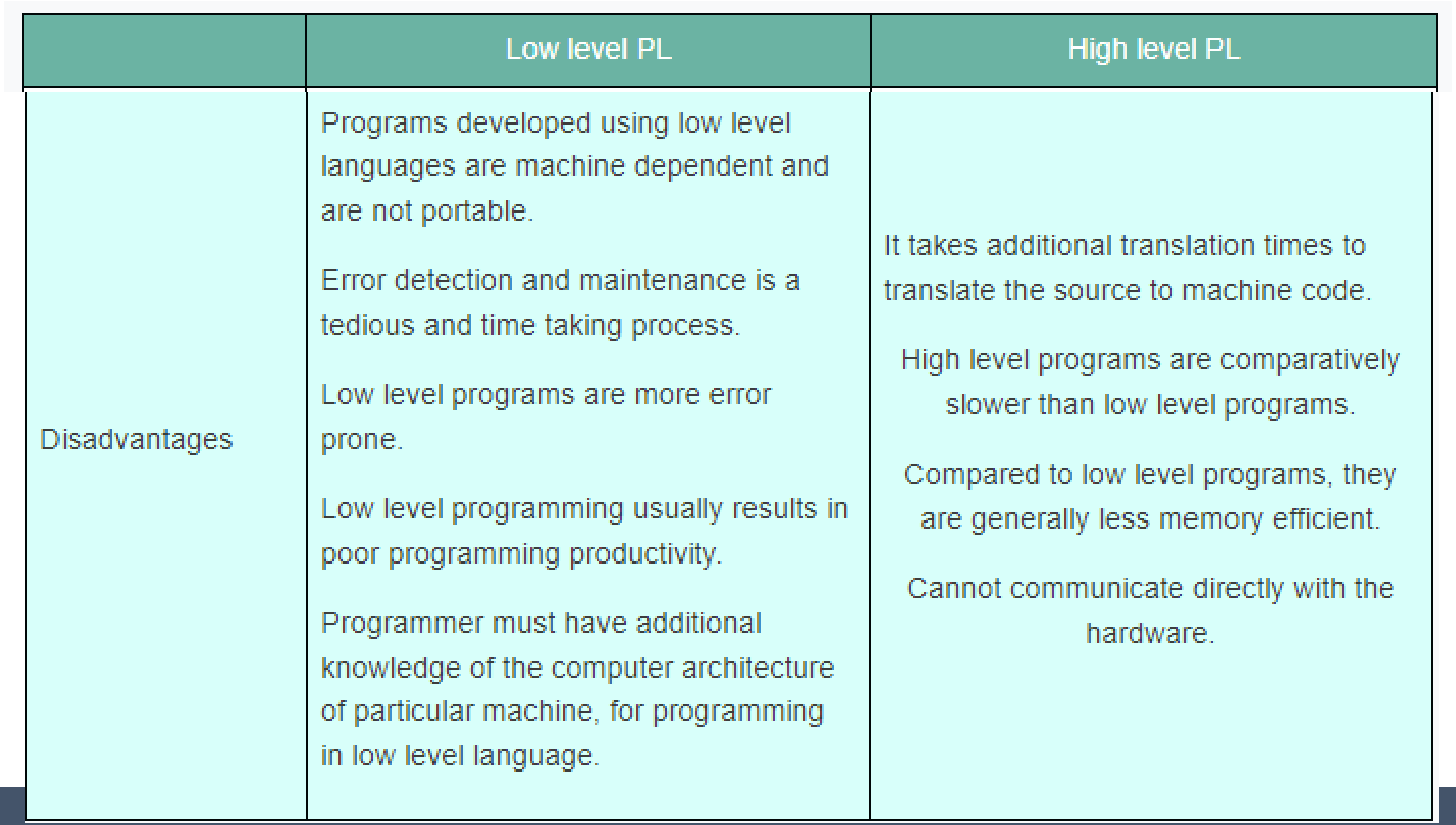


- **Imperative programming** is programming paradigm program describes a sequence of steps that change the state of the computer

- **Declarative programming** is a paradigm that expresses the desired result, not how to achieve it.



	Low level PL	High level PL
Advantages	Translated program requires less memory Write code that can be executed faster Total control over the code Can work directly on memory locations	Easier to modify as it uses English like statements Easier/faster to write code as it uses English like statements Easier to debug during development due to English like statements Portable code – not designed to run on just one type of machine



	Low level PL	High level PL
Disadvantages	Programs developed using low level languages are machine dependent and are not portable. Error detection and maintenance is a tedious and time taking process. Low level programs are more error prone. Low level programming usually results in poor programming productivity. Programmer must have additional knowledge of the computer architecture of particular machine, for programming in low level language.	It takes additional translation times to translate the source to machine code. High level programs are comparatively slower than low level programs. Compared to low level programs, they are generally less memory efficient. Cannot communicate directly with the hardware.

Any Questions???

Learning Objective:

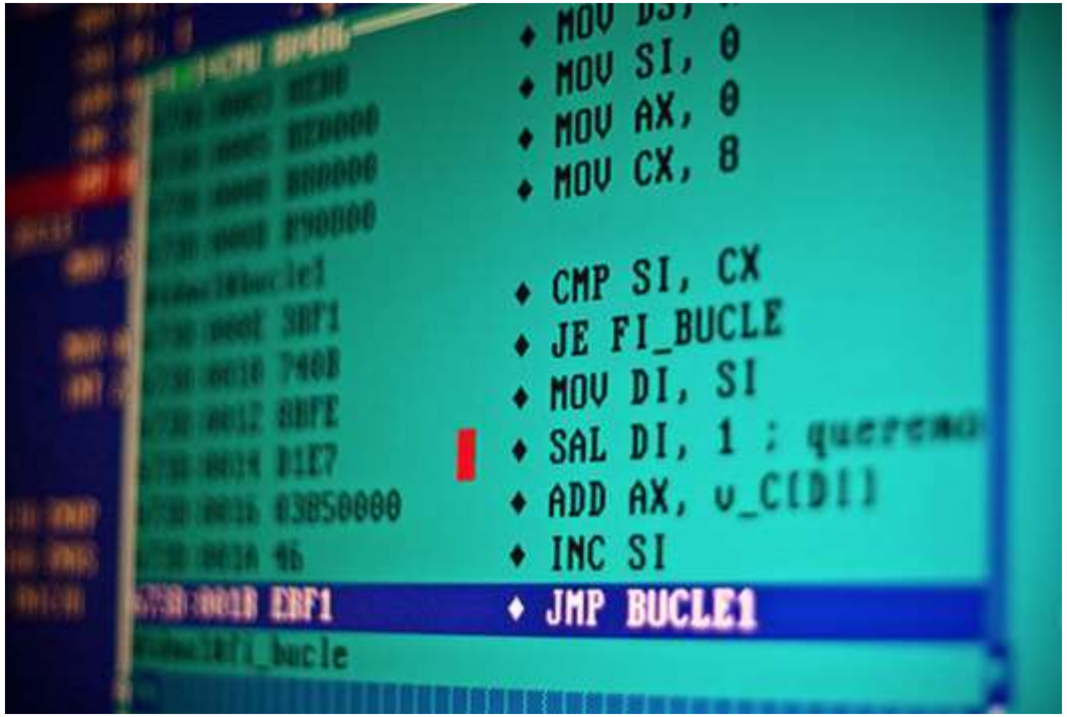
- ✓ advantages and disadvantages of compilers
- ✓ advantages and disadvantages of interpreters

✓ Success criteria:

- analyze the advantages and disadvantages of compilers and interpreters
- compare Language translators like compiler and interpreter

ASSEMBLERS

- Are platform specific
- Each assembly language instruction has a 1-to-1 relationship to a machine code instruction
- Translation is **fairly quick** and **straightforward**



```
MOV DS,
MOV SI, 0
MOV AX, 0
MOV CX, 8

CMP SI, CX
JE FI_BUCLE
MOV DI, SI
SAL DI, 1 ; queremo
ADD AX, v_C[DI]
INC SI
JMP BUCLE1
```

TYPES OF TRANSLATORS

Compilers

- Scans the entire program and translates the whole program at once
- Are platform specific
- Take a high level code as a source code
- **Check** the source code for any errors **line by line**
- Check the **entire** program **at ones**
- If the source code contains an error, it **will not be translated**
- Generates an intermediary **object code**
- Compiled programs can be run **without any other software present**

Interpreters

- Translates just one statement of the program at a time
- Check for errors **as they translate**
- Can be **partially translate** source code containing errors
- Both the program source code and the interpreter itself **must be present**
- This results in **poor protection** of the source code
- Does not generate an intermediary code

Learning objectives

Analyze a simple program written in the language of assembler

Use trace tables to find and verify the correctness of an algorithm

Success criteria

Understand the use of assembly language

Distinguish the difference between assembly language and others

Give definition for term "trace table"

Explain the purpose of using trace table

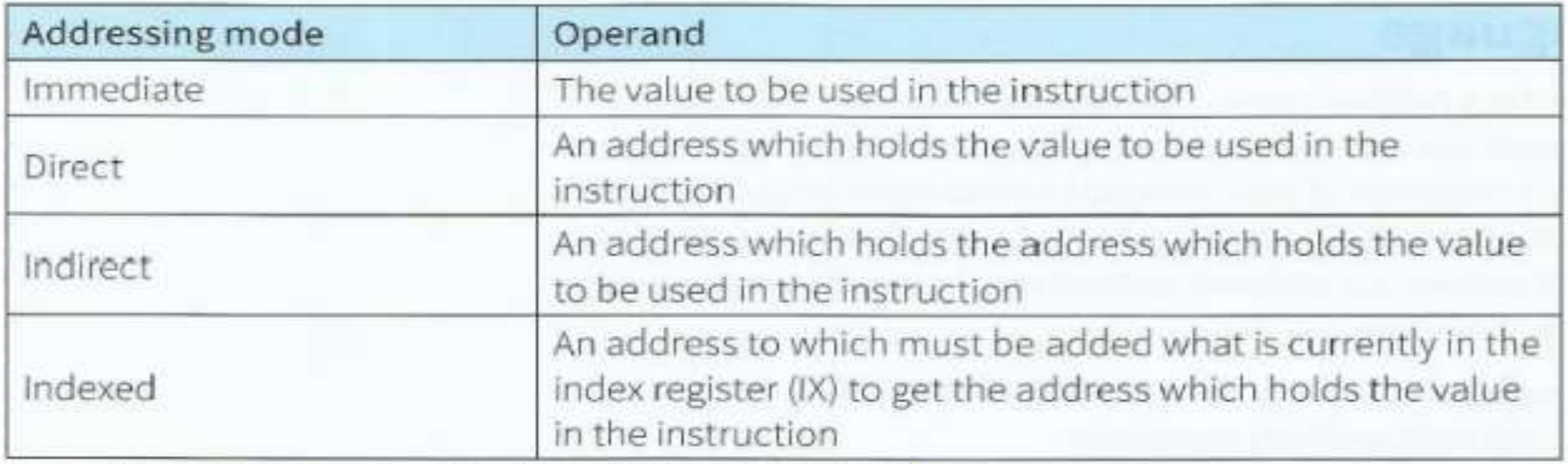
Build and fill trace table for checking results

Compare result of executed program and filled trace table

Addressing modes

When an instruction requires a value to be loaded into a register there are different ways of identifying the value.

These different ways are described as the 'addressing modes'. In Section 6.01, it was stated that, for our simple processor, two bits of the opcode in a machine code instruction would be used to define the addressing mode. This allows four different modes which are described in Table.



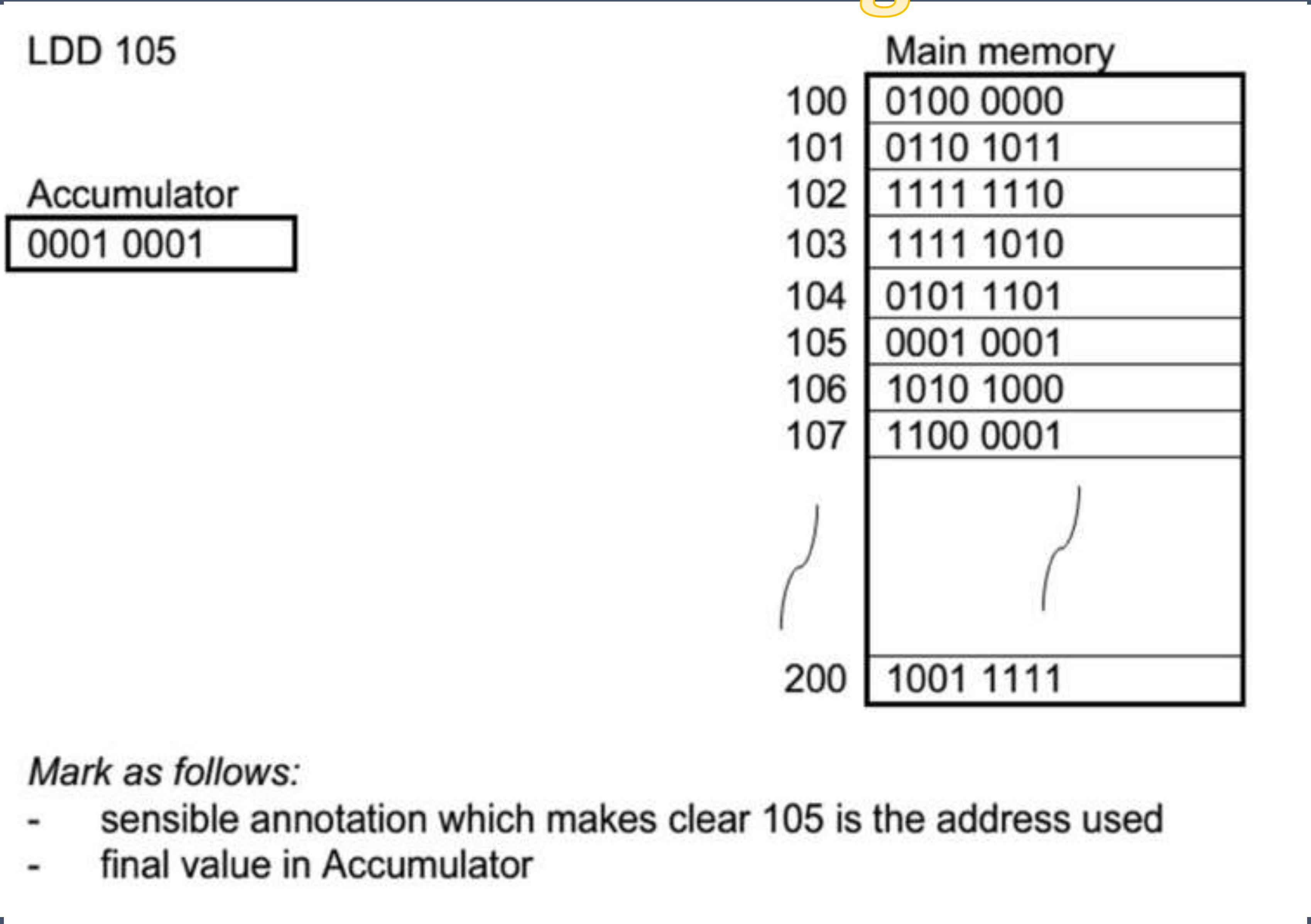
Addressing mode	Operand
Immediate	The value to be used in the instruction
Direct	An address which holds the value to be used in the instruction
Indirect	An address which holds the address which holds the value to be used in the instruction
Indexed	An address to which must be added what is currently in the index register (IX) to get the address which holds the value in the instruction

You might notice that some instructions use**”#”** and others don't
= number**, [No hash]** = address

Immediate addressing mode

Address	Instruction
101	LDA #12
102	

Direct addressing mode



LDD 105

Accumulator

0001 0001

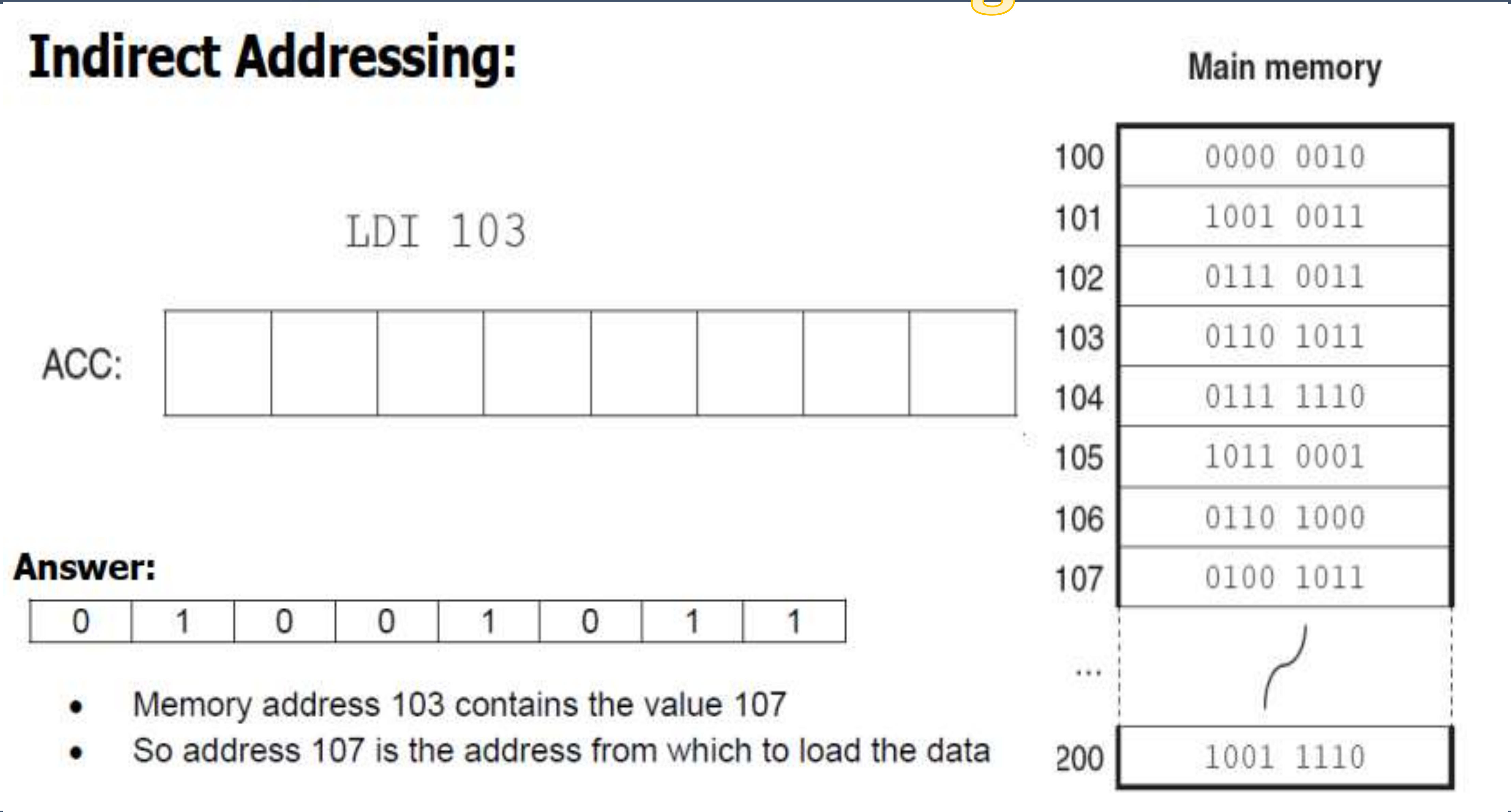
Main memory

100	0100 0000
101	0110 1011
102	1111 1110
103	1111 1010
104	0101 1101
105	0001 0001
106	1010 1000
107	1100 0001
200	1001 1111

Mark as follows:

- sensible annotation which makes clear 105 is the address used
- final value in Accumulator

Indirect addressing mode



Indirect Addressing:

LDI 103

ACC:

Answer:

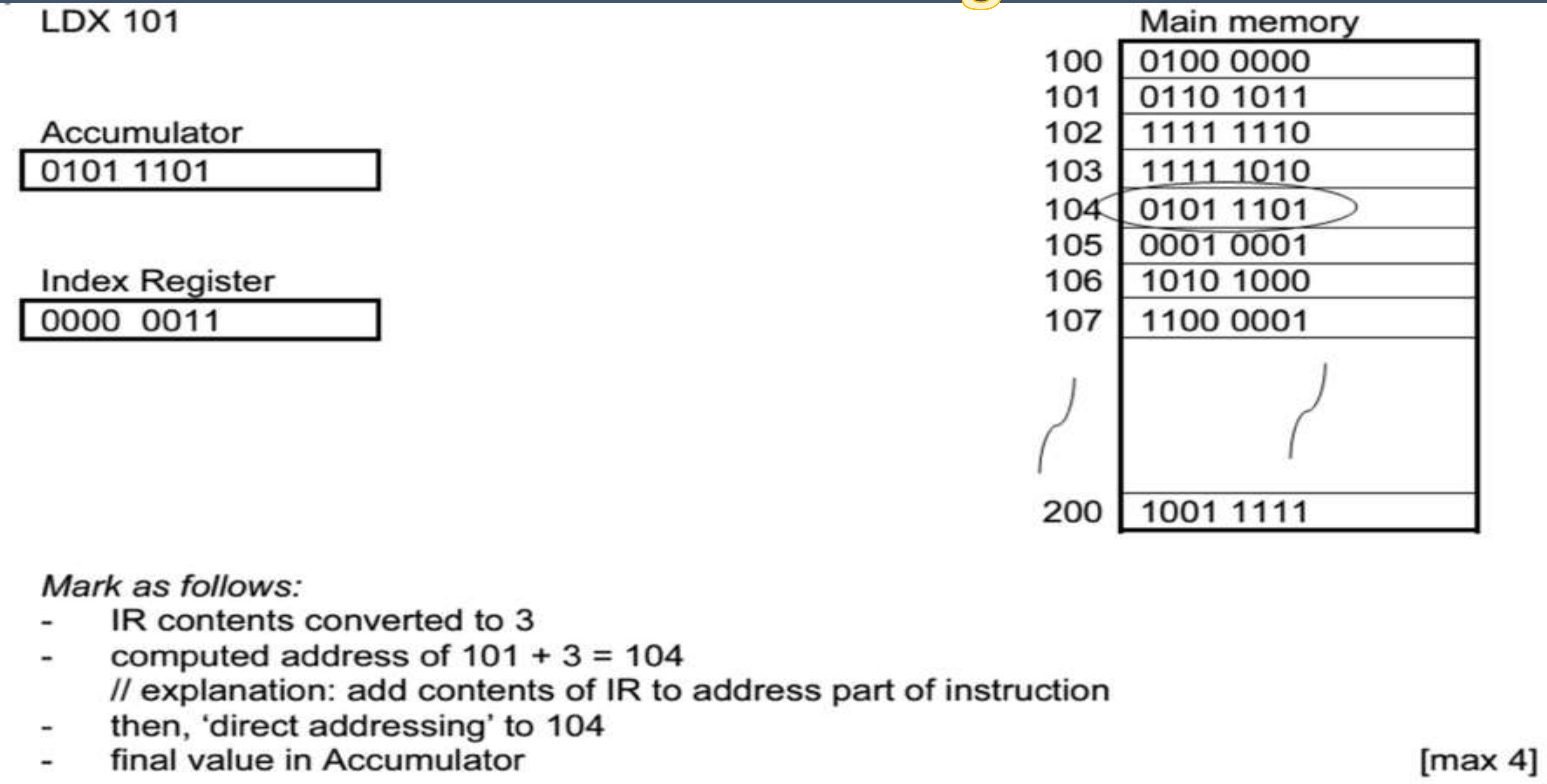
0	1	0	0	1	0	1	1

- Memory address 103 contains the value 107
- So address 107 is the address from which to load the data

Main memory

100	0000 0010
101	1001 0011
102	0111 0011
103	0110 1011
104	0111 1110
105	1011 0001
106	0110 1000
107	0100 1011
...	
200	1001 1110

Indexed addressing mode



LDX 101

Accumulator

0101 1101

Index Register

0000 0011

Main memory

100	0100 0000
101	0110 1011
102	1111 1110
103	1111 1010
104	0101 1101
105	0001 0001
106	1010 1000
107	1100 0001
200	1001 1111

Mark as follows:

- IR contents converted to 3
- computed address of 101 + 3 = 104
  // explanation: add contents of IR to address part of instruction
- then, 'direct addressing' to 104
- final value in Accumulator

[max 4]

Any Questions???

Task 1

Fill the trace table

Determine what problem this program code solves

```
1 LDA SECOND
2 SUB FIRST
3 BRP SECBIG
4 LDA FIRST
5 OUT
6 HLT
7 SECBIG LDA SECOND
8 OUT
9 HLT
10 FIRST DAT 5
11 SECOND DAT 7
```

Line	First	Second	Condition	Accumulator	Output

Assessment of criteria

	Descriptor	Score
11.5.1.4 use trace tables to find and verify the correctness of an algorithm	Correct filled Line column	1
	Correct filled First column	1
	Correct filled Second column	1
	Correct filled Condition column	1
	Correct filled Acc column	1
	Correct filled Output column	1

Answer

Line	First	Second	Condition	Accumulator	Output
	5	7			
1				7	
2				2	
3			BRP is TRUE		
7				7	
8					7

LMC Branch Instructions (for implementing loops)

-
  ```
  LDA ONE
  STA COUNT
  OUT
  LOOPTOP LDA COUNT
  ADD ONE
  OUT
  STA COUNT
  SUB TEN
  BRP ENDLOOP
  BRA LOOPTOP
  ENDLOOP HLT
  ONE DAT 001
  TEN DAT 010
  COUNT DAT
  ```

- ***The LOOPTOP identifier is the first instruction in the loop.***
- ***When the code in the loop has been executed, a BRANCH always instruction (e.g. BRA LOOPTOP) causes the LMC to "jump" back to the start of the loop so that the code section can be executed again.***

Task 2

```
LOOPTOP LDA COUNT
BRZ ENDLOOP
SUB ONE
STA COUNT
LDA TOTAL
ADD EIGHT
STA TOTAL
OUT
BRA LOOPTOP
ENDLOOP HLT
EIGHT DAT 008
COUNT DAT 003
ONE DAT 001
TOTAL DAT
```

1. Research this program and define result using trace table

2.Determine what problem this program code solves

Assessment of criteria

	Descriptor	Score
11.5.1.4 use trace tables to find and verify the correctness of an algorithm	Correct filled Count column	1
	Correct filled One column	1
	Correct filled Total column	1
	Correct filled Condition column	1
	Correct filled Acc column	1
	Correct filled Output column	1

Task 2

1. Research this program and define result using trace table

```
LOOPTOP LDA COUNT
BRZ ENDLOOP
SUB ONE
STA COUNT
LDA TOTAL
ADD EIGHT
STA TOTAL
OUT
BRA LOOPTOP
ENDLOOP HLT
EIGHT DAT 008
COUNT DAT 003
ONE DAT 001
TOTAL DAT
```

2.Determine what problem this program code solves

3.Run this program in LMC and check every step

SUMMATIVE ASSESMENT FOR UNIT

11.1B – Programming paradigms

20 min

Reflection