Introduction to Information Systems

– Understanding the digital world

4 Programming

Liang Zhao

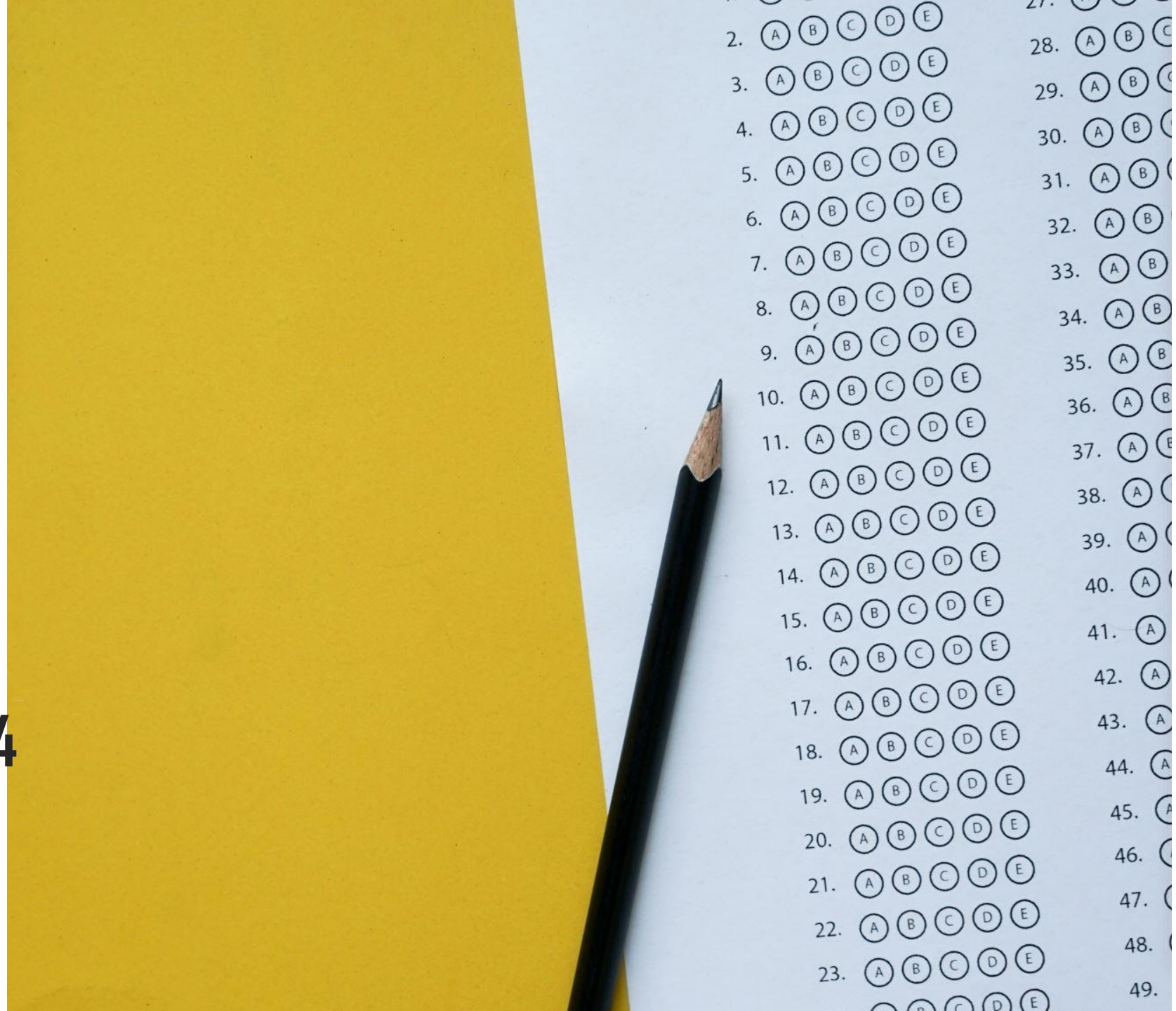
ILA, Doshisha University

12001102, Fall, 2024

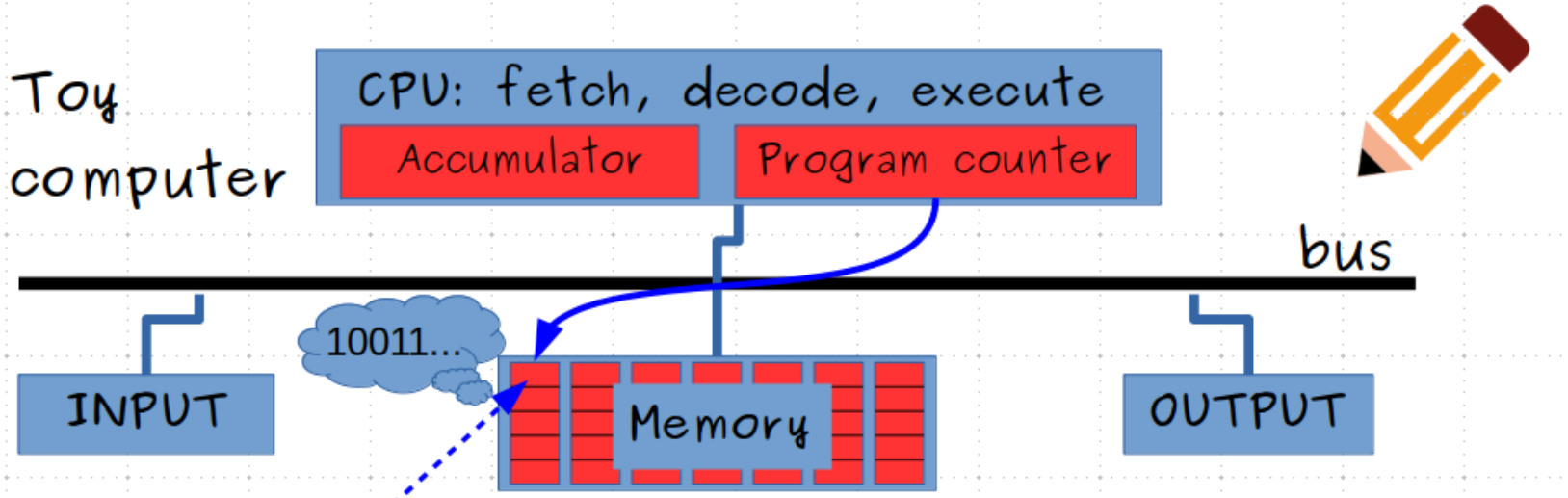
Today's schedule

- Review of Mini test #3 (5')
- Mini test #4 (15')
- Advanced coding with the Toy Machine (70')

On mini test #3 and mini test #4



Toy Machine & instructions



The programming language we are learning is called the assembly, which is almost the same as the machine language.

label	instruction	description
	get	get a number from keyboard into accumulator
L	print	print contents of accumulator
	load Val	load accumulator with Val (Val unchanged)
	store M	store contents of accumulator into memory location called M
	add Val	add Val to contents of accumulator (Val unchanged)
	sub Val	subtract Val from contents of accumulator (Val unchanged)
	goto L	go to instruction labeled L
	ifpos L	go to instruction labeled L if accumulator is >= zero
	ifzero L	go to instruction labeled L if accumulator is zero
	stop	stop running
M	Num	before program runs, set this memory location (called M) to Num

Straightforward answer to the last bonus task

Simply add A 99 times.

Toy Machine Simulator ->

http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html

```
_GET
_STORE A
_ADD A
...
_ADD A
_PRINT
_STOP
A
```

99 “ADD A”

A smart answer to the last bonus task

It uses the fact 100 = 10 x 10 to reduce the #additions from 99 to 18, which is 5.5 times faster than the straightforward one.

Toy Machine Simulator ->

http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html

```
1.  _GET
2.  _STORE A
3.  _ADD A
4.  _ADD A
5.  _ADD A
6.  _ADD A
7.  _ADD A
8.  _ADD A
9.  _ADD A
10. _ADD A
11. _ADD A
12. _STORE A
11. _ADD A
12. _ADD A
13. _ADD A
14. _ADD A
15. _ADD A
16. _ADD A
17. _ADD A
18. _ADD A
19. _ADD A
20. _PRINT
21. _STOP
22. A
```

A smarter answer to the last bonus task

It uses the fact 100 = 2 x 2 x 5 x 5
to reduce #additions to 10, which is 9.9 times faster than the straightforward one.

Toy Machine Simulator ->

http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html

1. ␣GET
2. ␣STORE A
3. ␣ADD A
4. ␣STORE A
5. ␣ADD A
6. ␣STORE A
7. ␣ADD A
8. ␣ADD A
9. ␣ADD A
10. ␣ADD A
11. ␣STORE A
12. ␣ADD A
13. ␣ADD A
14. ␣ADD A
15. ␣ADD A
16. ␣PRINT
17. ␣STOP
18. A

Another answer to the last bonus task

It uses the fact $100 = 3 \times 32 + 4$ to reduce #additions to 8, which is 12.4 times faster than the straightforward one.

Toy Machine Simulator ->

http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html

1. _GET
2. _STORE A
3. _ADD A
4. _STORE B
5. _ADD B
6. _STORE A
7. _ADD A
8. _STORE B
9. _ADD B
10. _STORE C
11. _ADD C
12. _STORE B
13. _ADD B
14. _ADD B
15. _ADD A
16. _PRINT
17. _STOP
18. A
19. B
20. C

This kind of approach is smart but tricky and limited.
The universal way is to use the "loop" structure.

Loop: repeat a task

A loop consists of a label and a jump.

IFZERO E: Jump to label E if the accumulator is 0.

GOTO L: Jump to label L (unconditionally).

SUB A: Subtract A from the accumulator

```
1. L␣GET
2. ␣SUB A
3. ␣IFZERO E
4. ␣GOTO L
5. E␣STOP
6. A␣2023
```

This one serves as a simple password checking program.

Another loop example

LOAD A: Load the content of A into the accumulator.

ADD 1: Add 1 to the accumulator.

For better understanding, use the single-step version of the toy machine.

http://www.cs.princeton.edu/courses/archive/fall18/cos109/toystep.html

1. L␣GET
2. ␣IFZERO E
3. ␣LOAD A
4. ␣ADD 1
5. ␣PRINT
6. ␣STORE A
7. ␣GOTO L
8. E␣STOP
9. A␣0

Input 0 to end it.

Solve the last bonus task with loop

This is the universal way to handle a loop with any number of repetition (e.g., 47, 739, or a number input by the user).

1. _GET
2. _STORE A
3. L_LOAD N
4. _IFZERO E
5. _SUB 1
6. _STORE N
7. _LOAD S
8. _ADD A
9. _STORE S
10. _GOTO L
11. E_LOAD S
12. _PRINT
13. _STOP
14. A
15. S_0
16. N_100

Ex. 1: Calculate a sum

This program calculates 1 + 2 + … + 100.
Consider how it can do that, then modify it to calculate 1 + 2 + … + 1000 please.

Toy Machine Simulator ->

http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html

```
1.  L␣LOAD N
2.  ␣IFZERO E
3.  ␣ADD S
4.  ␣STORE S
5.  ␣LOAD N
6.  ␣SUB 1
7.  ␣STORE N
8.  ␣GOTO L
9.  E␣LOAD S
10. ␣PRINT
11. ␣STOP
12. N 100
13. S 0
```

Ex. 2: Print a triangle

This program prints a triangle with a given height. Why? Confirm it with the toy machine.

Toy Machine Simulator ->

http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html

```
1.  _GET
2.  _STORE N
3.  L_LOAD N
4.  _IFZERO E
5.  _LOAD A
6.  _PRINT
7.  _ADD A
8.  _ADD A
9.  _ADD A
10. _ADD A
11. _STORE A
12. _ADD A
13. _ADD 8
14. _STORE A
15. _LOAD N
16. _SUB 1
17. _STORE N
18. _GOTO L
19. E_STOP
20. A_8
21. N
```

Summary

Three basic structures of a program

- Sequence (run code one by one)
- Conditional jump (a.k.a. selection)
- Loop

Any program can be decomposed to a combination of these three basic structures.

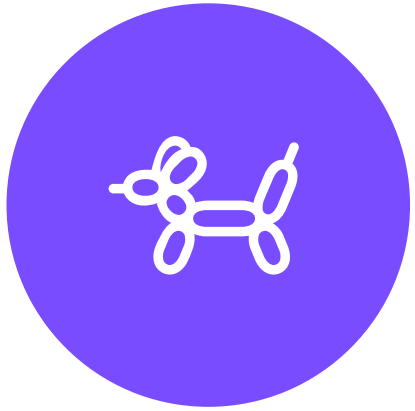
Coding tasks (3 points)

1. Write a program to calculate 1 + 2 + … + N for a number N input by the user (1 point).
2. Write a program that prints 1, 2, …, 99 in that order with loop (2 points).

Optional task (2 bonus points)

Write a program that prints an N x N square of “8” for a number N input by the user. Submit your answer to me by email before Oct 21.

Homework



WATCH THREE VIDEOS

YouTube -> Crash course -> Computer Science -> #11, #12, and #13
(You are not expected to understand everything)

READ CHAPTERS 4 & 5