



信息科学与技术学院

School of Information Science and Technology

CS 110

Computer Architecture

Datapath

Instructors:

Siting Liu & Chundong Wang

Course website: <https://toast->

[lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2024/index.html](https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2024/index.html)

School of Information Science and Technology (SIST)

ShanghaiTech University

2024/4/2

Administratives

- Lab 5 available, please prepare in advance! Lab 6 will also be released before the Qingming holiday.
- HW 3 available, ddl April 9th, start early!
- Proj1.1 ddl approaching, April 8th
- Proj1.2 released soon, ddl April 25th
- Future discussion (teaching center 301) schedule:
 - Friday (April 5th) no discussion (QingMing holiday).
 - April 7th (班) discussion on digital circuit basics by TA Yang Chao.
 - April 8th, mid-term review by Yang Chao.
 - After that, the same content for Friday and the next Monday.
- Labs on Thursday (April 4th) will be checked on Sunday (April 7th), 18:00-19:40. Please contact your TAs if you have further concerns.

Mid-term I

- Midterm I
 - April 11th **8:00 am - 10:00 am**
 - We start sharp at 8:00 am!
 - Arrive **7:45 am** to check-in (three classrooms likely and seat table will be determined on-site)
 - Arrive later than **8:30 am** will get 0 mark.
- Contents:
 - Everything till April 9th lecture
- Switch cell phones **off!!!** (not silent mode)
 - Put them in your bags.
- Bags in the front. On the table: nothing but pen, exam paper, 1 drink, 1 snack, **your student ID card** and your cheat sheet!

Mid-term I requirements

- You can bring a cheatsheet (**handwritten only**). **1-page A4, double-sided** (2-page for the mid-term II and 3-page for the final). Put it on your desk at exam. Cheatsheet that does not apply to the rules would be taken away.
- [Greencard](#) shown on the course website is provided with the exam paper.
- No other electronic devices are allowed!
 - No ear plugs, music, smartwatch, calculator, computer...
- Anybody touching any electronic device will **FAIL** the course!
- Anybody found cheating (copy your neighbors answers, additional material, ...) will **FAIL** the course!







COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE

 RISC- EDITION



MK
MORGAN KAUFMANN

DAVID A. PATTERSON
JOHN L. HENNESSY

Cheat Sheet

- **1 A4 Cheat Sheet** allowed (double sided)
 - Midterm II: 2 pages
 - Final: 3 pages
- Rules:
 - Hand-written – not printed/photocopied!
 - Your name in pinyin on the top!
 - Cheat Sheets not complying to this rule will be confiscated!

FUNCTIONS OF SEVERAL VARIABLES $z = f(x, y)$ $w = f(x, y, z)$ DOMAINS: Allowed (x, y) , (x, y, z) RANGES: z's, w's

LEVEL CURVES $z = f(x, y) = k = \text{CONST.}$
 CONTOUR MAPS (2-D)
 $w = f(x, y, z) = k = \text{CONST.}$
 SURFACE LAYERS (3-D)

FUNCTION OF n VARIABLES
 $z = f(x_1, x_2, \dots, x_n)$
 $f(x) = C \cdot x^n$
 1. As a function of n real variables x_1, x_2, \dots, x_n
 2. As a function of a single ρ -variable (x_1, \dots, x_n)
 3. As a function of a single vector var. $\vec{x} = (x_1, \dots, x_n)$

PARTIAL DERIVATIVES Derivatives w/ respect to one variable, while holding the other variables constant
 $z = f(x, y)$ NOTATIONS
 $f_x(x, y) = f_x = \frac{\partial z}{\partial x} = \frac{\partial f(x, y)}{\partial x}$
 $f_y(x, y) = f_y = \frac{\partial z}{\partial y} = \frac{\partial f(x, y)}{\partial y}$
 SAME HOLDS FOR FUNCTIONS OF MORE THAN TWO VARIABLES

SECOND PARTIAL DERIVATIVES
 $f_{xx} = \frac{\partial}{\partial x} \left(\frac{\partial z}{\partial x} \right) = \frac{\partial^2 z}{\partial x^2}$
 $f_{xy} = \frac{\partial}{\partial y} \left(\frac{\partial z}{\partial x} \right) = \frac{\partial^2 z}{\partial y \partial x}$
 $f_{yx} = \frac{\partial}{\partial x} \left(\frac{\partial z}{\partial y} \right) = \frac{\partial^2 z}{\partial x \partial y}$
 $f_{yy} = \frac{\partial}{\partial y} \left(\frac{\partial z}{\partial y} \right) = \frac{\partial^2 z}{\partial y^2}$

CLAIRAUT'S THEOREM
 IF f_{xy} AND f_{yx} ARE BOTH CONTINUOUS
 $f_{xy}(a, b) = f_{yx}(a, b)$

PARTIAL DIFF. EQS
 LAPLACE'S EQUATION
 $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ ETC...
 THE WAVE EQUATION
 $\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$

THE CHAIN RULE SINGLE VARIABLE $y = f(x)$, $x = g(t)$, $z = f(g(t))$
 $\frac{dz}{dt} = \frac{dz}{dx} \frac{dx}{dt}$
 CASE 1 $z = f(x, y)$, $x = g(t)$, $y = h(t)$ $z = f(g(t), h(t))$
 $\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}$
 CASE 2 $z = f(x, y)$, $x = g(s, t)$, $y = h(s, t)$ $z = f(g(s, t), h(s, t))$
 $\frac{dz}{ds} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial s} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial s}$
 $\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial t}$

CHAIN RULE: GENERAL VERSION
 $z = f(x_1, \dots, x_n)$
 $\frac{dz}{dt} = \sum_{i=1}^n \frac{\partial z}{\partial x_i} \frac{dx_i}{dt}$

IMPLICIT DIFFERENTIATION
 $F(x, y, z) = 0$
 $\frac{\partial z}{\partial x} = -\frac{F_x}{F_z}$
 $\frac{\partial z}{\partial y} = -\frac{F_y}{F_z}$

TANGENT PLANE TO A LEVEL SURFACE
 $F(x, y, z) = 0$
 $F_x(x_0, y_0, z_0) + F_y(x_0, y_0, z_0) + F_z(x_0, y_0, z_0) = 0$
 $\nabla F \cdot \vec{r} = 0$

NORMAL LINE TO A LEVEL SURFACE
 $\vec{n} = \nabla F = (F_x, F_y, F_z)$

MAXIMUM AND MINIMUM VALUES $z = f(x, y)$
 $f_x(a, b) = 0$, $f_y(a, b) = 0$ NECESSARY BUT NOT SUFFICIENT TO GUARANTEE A MAX. OR MIN.
 THEN APPLY THE 2ND DERIVATIVE TEST $D^2 f = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}$
 $D > 0$ LOCAL MIN.
 $D < 0$ LOCAL MAX.
 $D = 0$ SADDLEPT. NO CONCLUSION

FINDING ABSOLUTE MAX. AND MINS. FOR f ON A CLOSED BOUNDARY
 1. Find values of f at the critical points of f in D
 2. Find the extreme values of f on the boundary of D
 3. The largest value from 1, 2, is the ABS. MAX. the smallest is the ABS. MIN.

MAXIMIZING AND MINIMIZING Set of a function of two variables of the form $z = f(x, y)$ and then do the usual routine

E- δ DEFINITION OF CONTINUITY
 LET f BE A FUNCTION OF 2 VARIABLES DEFINED ON A DISK W/ CENTER (a, b) , EXCEPT POSSIBLY (a, b) . THEN $\lim_{(x, y) \rightarrow (a, b)} f(x, y) = L$ IF FOR EVERY $\epsilon > 0$, THERE IS A CORRESPONDING $\delta > 0$ ST. IF $(x, y) - L$ IS $< \epsilon$ whenever $|(x, y) - (a, b)| < \delta$

COMPOSITE FUNCTIONS OF CONTINUOUS FUNCTIONS ARE CONTINUOUS, AS ARE SUMS AND PRODUCTS

EQUATIONS OF TANGENT PLANES TO SURFACES
 $z = f(x, y)$ @ (x_0, y_0, z_0) EVALUATED AT A POINT
 $z - z_0 = f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$

TOTAL DIFFERENTIAL $(dy = f'(x) dx)$ SINGLE VARIABLE
 $dz = f_x(x, y) dx + f_y(x, y) dy = \frac{\partial z}{\partial x} dx + \frac{\partial z}{\partial y} dy$

INCREMENT $\Delta x, \Delta y, \Delta z$ DIFFERENTIALS dx, dy, dz
 FOR SMALL $\Delta x, \Delta y$ $\Delta z \approx dz$
 IF f_x AND f_y ARE CONTINUOUS $\Delta z \approx dz$
 (is change in height of surface Δz \approx change in height of the tangent plane dz)

THEOREM
 $\Delta z = f(a + \Delta x, b + \Delta y) - f(a, b)$
 $\Delta z = f_x(a, b) \Delta x + f_y(a, b) \Delta y + \epsilon_1 \Delta x + \epsilon_2 \Delta y$ where ϵ_1 and ϵ_2 are functions of Δx and Δy that approach 0 as $(\Delta x, \Delta y) \rightarrow (0, 0)$ DEF.

DEPENDENCY DIAGRAMS CASE 1, 2
 YOU CAN FIND DERIVATIVES FOR ALL THE FUNDAMENTAL VARIABLES

THE GRADIENT VECTOR $z = f(x, y)$ AT A POINT
 $\nabla f(x, y) = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right) = (f_x, f_y) = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right)$

DIRECTIONAL DERIVATIVES $D_{\vec{u}} f$, $\vec{u} = (a, b)$
 $D_{\vec{u}} f(x, y) = f_x(x, y) a + f_y(x, y) b$ SAME FOR 3 VARIABLES
 $D_{\vec{u}} f(x, y) = \nabla f(x, y) \cdot \vec{u}$
 $D_{\vec{u}} f$ MAX OCCURS WHEN ∇f IS IN THE SAME DIR. AS \vec{u}
 $D_{\vec{u}} f = \nabla f \cdot \vec{u} = |\nabla f| |\vec{u}| \cos \theta = |\nabla f| |\vec{u}| \cos 0$

THE GRADIENT VECTOR POINTS IN THE DIRECTION OF STEEPEST ASCENT OR DESCENT (ON A SURFACE)

THE GRADIENT VECTOR IS ORTHOGONAL TO THE LEVEL CURVES OF A SURFACE

TO FIND THE NORMAL (AND LATER TANGENT PLANE) TO A SURFACE, LET THAT SURFACE BE THE LEVEL SET OF SOME HIGHER DIMENSIONAL FUNCTION. THEN THE GRADIENT OF THE HIGHER D FUNCTION IS \perp TO YOUR SURFACE
 $x^2 + y^2 + z^2 = 1$ Let $w = x^2 + y^2 + z^2 - 1$
 $x^2 + y^2 + z^2 = 1$ IS THE LEVEL SET $w = 0$
 SO $\nabla w = (2x, 2y, 2z)$ IS A NORMAL VECTOR TO THE 3-D SPHERE $x^2 + y^2 + z^2 = 1$

WAVES
 DISTANCE: $\Delta x = v \Delta t$
 PERIOD: $T = \frac{1}{f}$
 WAVELENGTH: $\lambda = v T = \frac{v}{f}$
 VELOCITY: $v = \lambda f$
 KEINETIC ENERGY: $\frac{1}{2} \rho v^2 A^2 \omega^2$
 GRAVITATIONAL POTENTIAL: mgh
 POWER: $P = F \cdot v = \tau v$

INTERFERENCE
 CONSTRUCTIVE: $\Delta \phi = 0$
 DESTRUCTIVE: $\Delta \phi = \pi$

DIFFRACTION IS THE BENDING OF LIGHT AROUND OBSTACLES AND THROUGH APERTURES.

LAW OF REFLECTION
 THE ANGLE OF INCIDENCE IS EQUAL TO THE ANGLE OF REFLECTION. THE INCIDENT AND REFLECTED BEAMS, AND THE NORMAL, ALL LIE ON THE SAME PLANE.

REFRACTION IS THE BENDING OF THE PATH OF LIGHT DUE TO A CHANGE IN SPEED AS IT ENTERS A MEDIUM OF DIFFERENT OPTICAL DENSITY.

SPASQUE: doesn't allow light to pass thru
TRANSPARENT: allows most light to pass
TRANSLUCENT: allows light through but distorts its path to blur the image.
 No material can allow 100% of incident light to pass through.

REFRACTION IS THE BENDING OF THE PATH OF LIGHT DUE TO A CHANGE IN SPEED AS IT ENTERS A MEDIUM OF DIFFERENT OPTICAL DENSITY.

STRAW: slower medium = refracted towards faster medium = retroreflected away from less dense

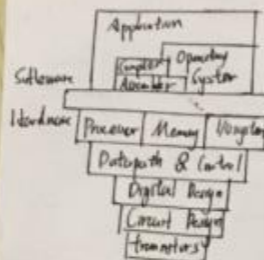
NEWTON'S LAWS
 1. A body's speed remains constant if the net force acting on it is zero (balanced).
 Acceleration is directly proportional to net force and inversely proportional to mass. Forces exist in action/reaction pairs.

MEMBER THAT WHEN GRAVITY IS WINNING AGAINST THE SUM OF FORCES, IT WOULD BE FACTORED AS A NEGATIVE

CONVERSION
 $m/s \xrightarrow{\times 3.6} km/h$
 $km/h \xrightarrow{\div 3.6} m/s$

OBSCURE
 SPECTRAL CLASSIFICATION SCHEME is based on temperature
 COOLER STARS HAVE MORE COMPLEX ELEMENTARY MAGNITUDE
 APPARENT MAGNITUDE: magnitude it has as seen by an observer on earth
 ABSOLUTE MAGNITUDE: magnitude it has if it were 10 parsecs from earth.
 DIFFERENCE OF 1 MAGNITUDE = 2.512x BRIGHTER.
 LUMINOSITY: measure of the total amount of energy radiated by a star/second.
 THE HOTTER THE STAR THE GREATER ITS LUMINOSITY.
 THE BIGGER THE STAR (SURFACE, RADIUS) THE MORE LUMINOUS.
 CEPHEIDS ARE STARS THAT VARY THEIR BRIGHTNESS OVER PERIODS OF DAYS.
 AMPLITUDE RANGE: 0.5 to 2.5 magnitudes.
 THE GREATER THE PERIOD, THE GREATER THE LUMINOSITY OF THE STAR.
 THE SUN IS AN ORANGE DWARF STAR.
FUSION
 $4H^+ \rightarrow He^2+ + 2e^- + \text{energy}$
 ENERGY IS RELEASED WHEN TWO LIGHT ATOMS FUSE TO FORM A HEAVIER NUCLEUS. THEY RELEASE ENERGY WHEN THEY RELEASE ENERGY WHEN THEY RELEASE ENERGY.
 ON THE MAIN SEQUENCE USE UP HYDROGEN.
 STAR HEATS, SHRINKS THEN EXPANDS.
 EARTH \rightarrow YELLOW DWARF \rightarrow TODAY \rightarrow RED GIANT

Old School Machine Structures



When C program starts
 - C available part is loaded into memory by OS (copying system)
 - OS set up stack, then calls into main function
 - Run for first initialises memory and other libraries
 - Then will go to procedure round main()

Valid Pointer Arithmetic
 - Add an integer to a pointer
 - Subtract 2 pointers (in the same array)
 - Compare two pointers (c.c., ...)
 - Compare pointer to null
 - Add two pointers / multiply

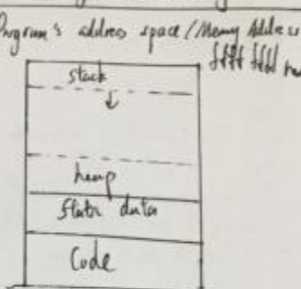
Six Fundamental Steps in Calling a Function
 1. Put parameters in a place where function can access them
 2. Transfer control to function
 3. Acquire local storage resources needed for function
 4. Perform declared task of the function
 5. Put result value in a place where calling code can access it and restore any registers you used
 6. Return control to part of origin, since a function can be called from several parts in a program.

New School Machine Structures

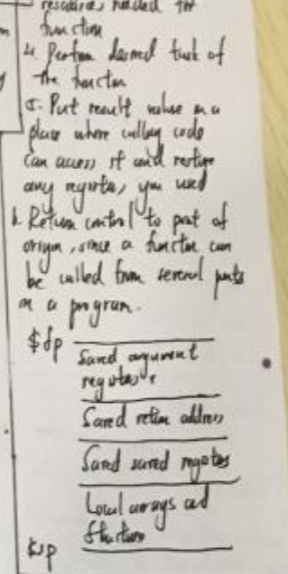
- Parallel Requests
- Parallel Threads/Threads
- Parallel Data
- Hardware descriptors
- Great Ideas in Computer Architecture
 - Abstraction
 - Layers of Representation/Interpretation
 - Moore's Law
 - Designing through trends
 - Principle of Locality
 - Memory Hierarchy
 - Parallelism
 - Performance Measurement and Improvement
 - Dependability via Redundancy

- 优先级: 从上至下依次进行
 1. 行 数值
 2. 左 右
 3. 左 右
 4. 左 右
 5. 左 右
 6. 左 右
 7. 左 右
 8. 左 右
 9. 左 右

int main (int argc, char * argv[])
 - argc contains the number of strings on the command line
 - argv is a pointer to an array containing the arguments as strings



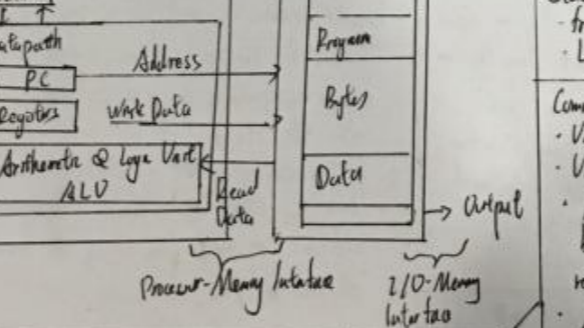
Stack: local variables, needs function, grow downwards
 Heap: space requested for dynamic data via malloc, grows dynamically
 Static data: variable declared outside functions, loaded into program starts, can be modified
 Code: loaded when program starts, cannot change



Two's-Complement Representation

- treats 0 as positive
 - 32-bit word represents 2^{32} into from -2^{31} to $2^{31}-1$

Components of a Computer



Stack
 - free when function returns
 - Last in first out

RISC format: used for instructions with immediate, for and or and branches (big and big)

func.c -> cpp -> fo.o -> compiler
 - cpp replaces comments with a single space
 - cpp comments begins with #

Common Memory Problems
 - Very uninitialised values
 - Using memory that you don't own
 - Improper use of free/malloc by messing with the pointer handle returned by malloc/calloc
 - Memory leaks

Opcode rs rt rd shamt funct
 2 format: 5-bit field only represents number up to 31.
 If instruction has immediate, then use or not 2 registers used for (m, sw, beq, bne) branches and with immediate

- 13. ? 左到右
- 14. += -= *= /= %= 左到右
- 15. , 左到右

Levels of Representation / Interpretation
 High level language
 Assembly Language
 Machine Language
 Machine / Interpreter
 Hardware Architecture Description
 Architecture / Implementation
 Logic Level Description

Declaring with larger immediate
 Load Upper Immediate lui
 \$t0 \$t0 0xABABC0D0
 \$t0 \$t0 0xABAB
 \$t0 \$t0 0xAB
 \$t0 \$t0 0xAB

Address (CPU) A3-0
 Address (CPU) A3-1
 Address (CPU) A3-2
 Address (CPU) A3-3
 Address (CPU) A3-4
 Address (CPU) A3-5
 Address (CPU) A3-6
 Address (CPU) A3-7
 Address (CPU) A3-8
 Address (CPU) A3-9
 Address (CPU) A3-10
 Address (CPU) A3-11
 Address (CPU) A3-12
 Address (CPU) A3-13
 Address (CPU) A3-14
 Address (CPU) A3-15
 Address (CPU) A3-16
 Address (CPU) A3-17
 Address (CPU) A3-18
 Address (CPU) A3-19
 Address (CPU) A3-20
 Address (CPU) A3-21
 Address (CPU) A3-22
 Address (CPU) A3-23
 Address (CPU) A3-24
 Address (CPU) A3-25
 Address (CPU) A3-26
 Address (CPU) A3-27
 Address (CPU) A3-28
 Address (CPU) A3-29
 Address (CPU) A3-30
 Address (CPU) A3-31

int division A3-28
 int multiple A3-27
 Macro (long) B2-8
 performance of CPU A3-25
 represent I/O B2-3
 represent denoms B2-5
 represent float IEEE754 B2-2
 represent floating point FP B2-1
 represent fraction A3-24
 represent bit number N,N B2-4
 represent scientific notation A3-23
 system performance evaluation A3-26

A-1

1. Abstraction: Ask Intel don't need to know how
 Moore's Law: 2X transistor/cip every 2 years
 Principle of Locality: fast-expensive \rightarrow fast
 Parallelism: do many things at once \rightarrow fast
 Performance measurement/Improvement:
 explicit: latency/throughput/speed/run time
 Reliability via Redundancy: Have things
 Similar use, one fail, but we can use other instead

2. Hierarchical digits 10, 12, ... A,B,C,D,E,F

3. Overflow (Two's complement)

4. Components of computer

5. Connection

6. Advantages

7. Disadvantages

8. Stack

9. Memory

10. Cache

11. Branch

12. Branch prediction

13. Branch misprediction

14. Branch misprediction penalty

15. Branch misprediction penalty

16. Branch misprediction penalty

17. Branch misprediction penalty

18. Branch misprediction penalty

19. Branch misprediction penalty

20. Branch misprediction penalty

21. Branch misprediction penalty

22. Branch misprediction penalty

23. Branch misprediction penalty

24. Branch misprediction penalty

25. Branch misprediction penalty

26. Branch misprediction penalty

27. Branch misprediction penalty

28. Branch misprediction penalty

29. Branch misprediction penalty

30. Branch misprediction penalty

A-2

1. Abstraction: Ask Intel don't need to know how
 Moore's Law: 2X transistor/cip every 2 years
 Principle of Locality: fast-expensive \rightarrow fast
 Parallelism: do many things at once \rightarrow fast
 Performance measurement/Improvement:
 explicit: latency/throughput/speed/run time
 Reliability via Redundancy: Have things
 Similar use, one fail, but we can use other instead

2. Hierarchical digits 10, 12, ... A,B,C,D,E,F

3. Overflow (Two's complement)

4. Components of computer

5. Connection

6. Advantages

7. Disadvantages

8. Stack

9. Memory

10. Cache

11. Branch

12. Branch prediction

13. Branch misprediction

14. Branch misprediction penalty

15. Branch misprediction penalty

16. Branch misprediction penalty

17. Branch misprediction penalty

18. Branch misprediction penalty

19. Branch misprediction penalty

20. Branch misprediction penalty

21. Branch misprediction penalty

22. Branch misprediction penalty

23. Branch misprediction penalty

24. Branch misprediction penalty

25. Branch misprediction penalty

26. Branch misprediction penalty

27. Branch misprediction penalty

28. Branch misprediction penalty

29. Branch misprediction penalty

30. Branch misprediction penalty

A-3

1. Abstraction: Ask Intel don't need to know how
 Moore's Law: 2X transistor/cip every 2 years
 Principle of Locality: fast-expensive \rightarrow fast
 Parallelism: do many things at once \rightarrow fast
 Performance measurement/Improvement:
 explicit: latency/throughput/speed/run time
 Reliability via Redundancy: Have things
 Similar use, one fail, but we can use other instead

2. Hierarchical digits 10, 12, ... A,B,C,D,E,F

3. Overflow (Two's complement)

4. Components of computer

5. Connection

6. Advantages

7. Disadvantages

8. Stack

9. Memory

10. Cache

11. Branch

12. Branch prediction

13. Branch misprediction

14. Branch misprediction penalty

15. Branch misprediction penalty

16. Branch misprediction penalty

17. Branch misprediction penalty

18. Branch misprediction penalty

19. Branch misprediction penalty

20. Branch misprediction penalty

21. Branch misprediction penalty

22. Branch misprediction penalty

23. Branch misprediction penalty

24. Branch misprediction penalty

25. Branch misprediction penalty

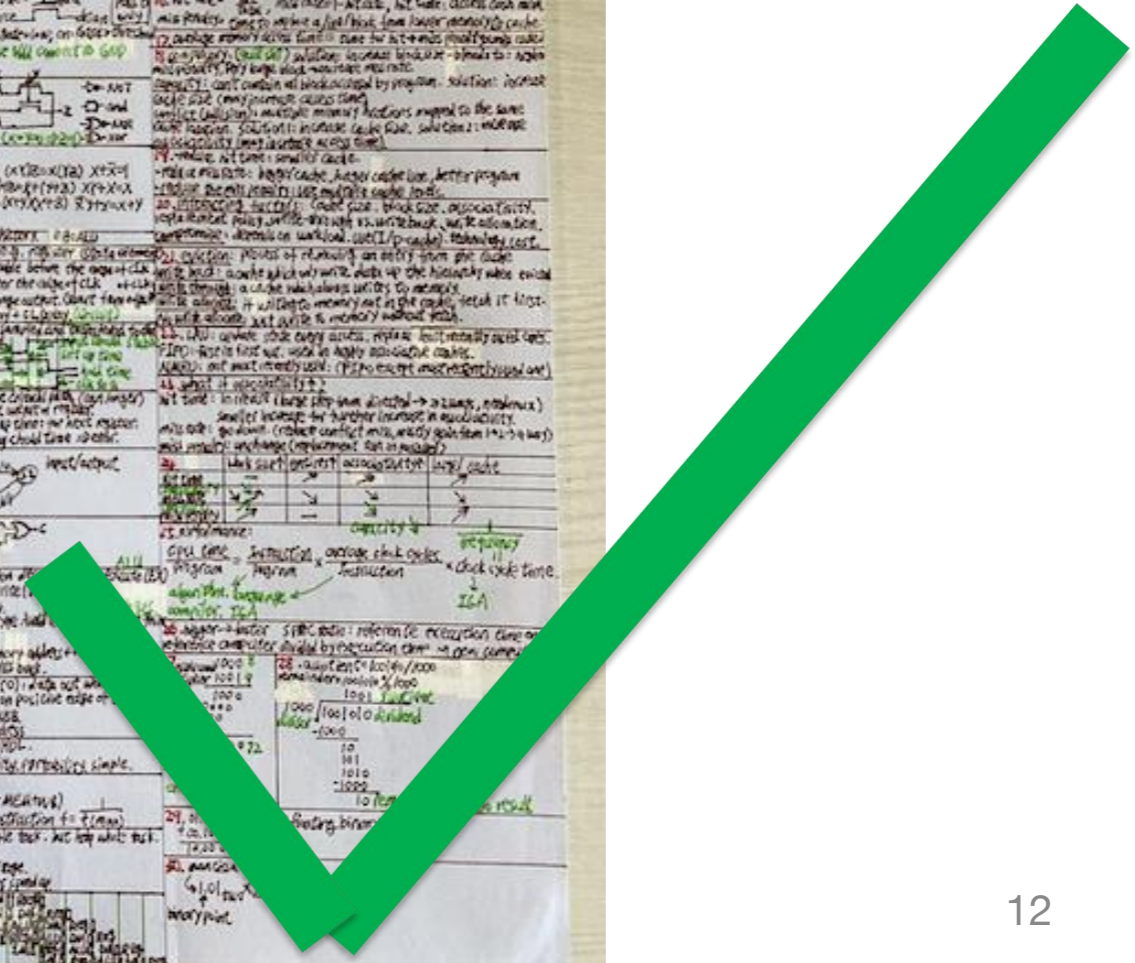
26. Branch misprediction penalty

27. Branch misprediction penalty

28. Branch misprediction penalty

29. Branch misprediction penalty

30. Branch misprediction penalty



1. VISIT: <http://computer-forensics11.sans.org/community/downloads>

2. BOOT SIFT VM

Login: sansforensics
Password: forensics

3. ELEVATE PRIVS

```
$ sudo su
```

4. CONNECT IMAGE TO SIFT

Plug hard drive to physical host and attach to SIFT VM

5. HARD DRIVE MOUNTING (if you are using log2timeline-sift and Single DD you can skip to 7-A)

SINGLE OR SPLIT IMAGE (2 options):

```
# mount_ewf.py image.E01 /mnt/ewf
```

```
# mount_ewf image.E01 /mnt/ewf/
```

```
# mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### /mnt/ewf/<image> /mnt/windows_mount/
```

MOUNT TO MOUNT POINT

SINGLE IMAGE

```
# mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### image.dd /mnt/windows_mount/
```

SPLIT IMAGE (2 step process)

```
# affuse image.001 /mnt/aff
```

```
# mount -t ntfs-3g -o loop,ro,show_sys_files,streams_interface=windows,offset=#### /mnt/aff/<image> /mnt/windows_mount/
```

6. log2timeline default timezone is set to examiner local host. To change use -z [TIMEZONE] option. To list all available timezones: # log2timeline -z list

7-A: AUTOMATED SUPER TIMELINE CREATION

```
log2timeline-sift -o -z [TIMEZONE] -p [PARTITION #] -i [IMAGE FILE]
```

DISK IMAGE (prompt for partition, mount, and run):

```
XP # log2timeline-sift -z EST5EDT -i image
```

```
WIN7 # log2timeline-sift -win7 -z EST5EDT -i image
```

FOR PARTITION (mount and run using all applicable plugins):

```
XP # log2timeline-sift -z EST5EDT -p 0 -i partition
```

```
WIN7 # log2timeline-sift -win7 -z EST5EDT -i partition
```

OTHER USAGE EXAMPLES:

```
# log2timeline -f list
Run log2timeline using only specific plugins:
# log2timeline-sift -p prefetch -z EST5EDT -i image.dd
Help (man page)
# log2timeline -h
```

8. CSV FILE OUTPUT (/cases/timeline-output-folder)

```
-date: Date of the event, in the format of MM/DD/YYYY
-time: Time of day, expressed in a 24h format, HH:MM:SS
-tz: Timezone: the timezone that was used to call the tool with.
-source: MACB meaning of the fields, comp w/ mactime format.
-sourcetype: Source short name (i.e. registry entries are REG)
-desc: Desc of the source ("Internet Explorer" instead of WEBHIST)
-type: Timestamp type (i.e. "Last Accessed", "Last Written")
-user: Username associated with the entry, if one is available.
-host: Hostname associated with the entry, if one is available.
-short: Contains less text than the full description field.
-desc: where majority info is stored, the actual parsed desc of the entry.
-version: Version number of the timestamp object.
-filename: Filename with the full path that contained the entry
-inode: inode number of the file being parsed.
-notes: Some input modules insert additional information in the form of a note, which comes here. Or it can be used during the review.
-format: Input module name used to parse the file.
-extra: Additional information parsed is joined together and put here.
```

7-B: MANUAL "MICRO" TIMELINE CREATION

```
log2timeline-sift -o -z [TIMEZONE] -p [PARTITION #] -i [IMAGE FILE] [-f FORMAT] [-z TIMEZONE] [-o OUTPUT MODULE] [-w LOG_FILE/LOG_DIR [-] [FORMAT FILE OPTIONS]]
```

EXTRACT METADATA (using log2timeline or fls)

```
# log2timeline -f mft -o mactime -r -z EST5EDT -w mft.body /mnt/volume/
OR Extract metadata using Sleuthkit:
# fls -m "" -o fls.body image.dd > fls.body
Convert body file format to mactime format w/ mactime:
# mactime -b fls.body -d > log2timeline.csv
```

ARTIFACTS (run l2l on mounted file system with plugins recursively)

```
Extract artifacts w/ log2timeline and run on mounted file system:
# log2timeline -f firefox3,chrome -o mactime -r -z EST5EDT -w web.body /mnt/volume/
Convert body file format to CSV format w/ mactime:
# mactime -b log2timeline.body -d > log2timeline.csv
```

9. FILTER TIMELINE

```
Filter timeline with date range to include only:
l2t_process -b timeline.csv MM-DD-YYYY..MM-DD-YYYY > filtered.csv
Filter timeline with keyword list (one term per line in keywords.txt):
l2t_process -b timeline.csv -k keywords.txt > filtered.csv
What sources are in your timeline?
awk -F , '{print $6;}' timeline.csv | grep -v sourcetype | sort | uniq
Find all LNK files that reference E Drive
grep "Shortcut LNK" timeline.csv | grep "E:"
Find MountPoints2 entries that reference E Drive
grep "MountPoints2 key" timeline.csv | grep "E drive"
grep USB timeline.csv | grep "SetupAPILog"
```

File System	M	A	C	B
Ext2/3	Modified	Accessed	Changed	N/A
FAT	Written	Accessed	N/A	Created
NTFS	File Modified	Accessed	MFT Modified	Created
UFS	Modified	Accessed	Changed	N/A

THE PURPOSE OF THIS REFERENCE GUIDE IS TO WALK THROUGH THE PROCESS OF BOOTING THE SIFT WORKSTATION, CREATING A TIMELINE ("SUPER" OR "MICRO") AND REVIEWING IT.

HOW TO CALCULATE THE OFFSET FOR MOUNTING

1. Run mmtools to query partition layout # mmtools image.E01
2. Identify partition and byte offset
3. (Partition byte offset) x (bytes per sector) = offset ##### to use!
Example: 63 X 512 = 32256

Note: If needed, repeat for each partition. Make new mount point: # mkdir /mnt/windows_mount2/

7-A & 7-B

HELP? OPTIONS? USAGE?

```
log2timeline -help
Log2timeline-sift -help
L2t_process -help
```

OTHER log2timeline OUTPUT FORMATS

Note: CSV is Default Output

- BeeDocs - Mac OS X visualization tool
- CEF - Common Event Format - ArcSight
- CFTL - XML file- CyberForensics TimeLab visualization tool
- CSV - comma separated value file
- Mactime - Both older and newer version of the format supported for use by TSK's mactime
- SIMILE - XML file - SIMILE timeline visualization widget
- SQLite - SQLite database
- TLN - Tab Delimited File
- TLN - Format used by some of H Carvey tools, expressed as a ASCII output
- TLNX - Format used by some of H Carvey tools, expressed as a XML document

10. CONNECT TO SIFT

- ✓ Settings -> SETTINGS -> OPTIONS -> Shared Folders -> Always Enabled (Check)
- ✓ 2. SIFT Desktop > VMware-Shared-Drive
- ✓ Access from a Win Machine \\SIFTWORKSTATION

11. REVIEW TIMELINE

- Review timelines using:
- Open, Soft, Filter with Excel
 - Import into SPLUNK
 - SIMILE
 - Tapestry

- log2timeline PARSING PLUGINS
- apache2_error - Apache2 error log file
 - chrome - Chrome history file
 - encase_dirlisting - CSV file that is exported from encase
 - evt - Windows 2k/XP/2k3 Event Log
 - evtx - Windows Event Log File (EVTX)
 - exif - Metadata information from files using ExifTool
 - ff_bookmark - Firefox bookmark file
 - firefox2 - Firefox 2 browser history
 - firefox3 - Firefox 3 history file
 - ftk_dirlisting - CSV file that is exported from FTK Imager (dirlisting)
 - generic_linux - Generic Linux logs that start with MMM DD HH:MM:SS
 - iehistory - index.dat file containing IE history
 - iis - IIS W3C log file
 - isatxt - ISA text export log file
 - jp_ntfs_change - CSV output file from JP (NTFS Change log)
 - mactime - Body file in the mactime format
 - mcafee - Log file
 - mft - NTFS MFT file
 - mssql_errlog - ERRORLOG file produced by MS SQL server
 - ntuser - NTUSER.DAT registry file
 - opera - Opera's global history file
 - oxml - OpenXML document pcap
 - pcap - PCAP file
 - pdf - Available PDF document metadata
 - prefetch - Prefetch directory
 - recycler - Recycle bin directory
 - restore 0.9 - Restore point directory
 - safari - Safari History.plist file
 - sam - SAM registry file
 - security - SECURITY registry file
 - setupapi - SetupAPI log file in Windows XP
 - skype_sql - Skype database
 - software - SOFTWARE registry file
 - sol - .sol (LSO) or a Flash cookie file
 - squid - Squid access log (http_emulate off)
 - syslog - Linux Syslog log file
 - system - SYSTEM registry file
 - tlm - Body file in the TLN format
 - volatility - Volatility output files (psscans2, sockscans2, ...)
 - win_link - Windows shortcut file (or a link file)
 - wmiprov - wmiprov log file
 - xpfirewall - XP Firewall log

List plugins # log2timeline -f list
...HELP EXPAND THIS LIST. BUILD PLUGINS!!!

BY DAVID NIDES (12/16/2011) ★★
TWITTER: @DAVNADS
BLOG: DAVNADS.BLOGSPOT.COM
EMAIL: DNIDES@KPMG.COM
CREDITS TO: ED GOINGS, ROB LEWIS, KRISTINN GUDJONSSON, KPMG & ...

KEY

- Red text - image/source
- Blue text - mount point
- Purple text - output file
- Green text - log2timeline plugins
- Brown text - Timezone

Outline

- Useful building blocks
 - ALU design
 - Register file
 - Memory considerations
- Datapath
- Design of the controller

Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

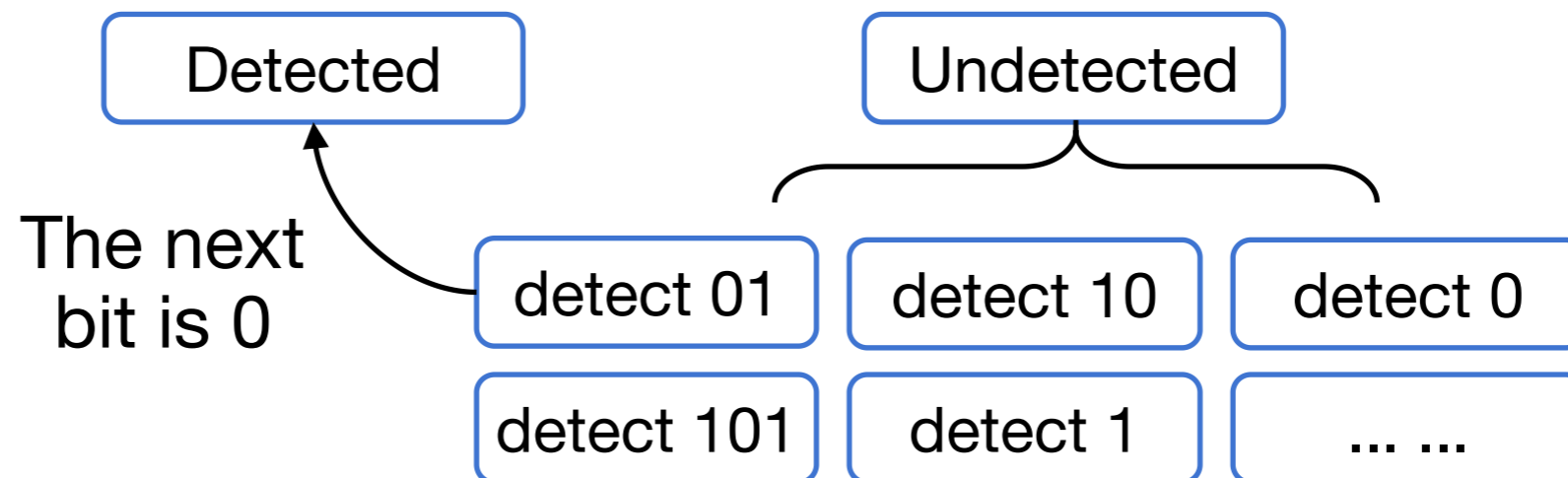
- Step 1: Draw finite state machine of the desired function (we ignore the initialization)
- Step 2: Define/assign binary numbers to represent the states, the inputs and the outputs
- Step 3: Write down the truth table (enumerate input/previous state (and current state) and their corresponding current state (and output))
- Step 4: Use template and decide the combinational block for state transition and output logic

Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function

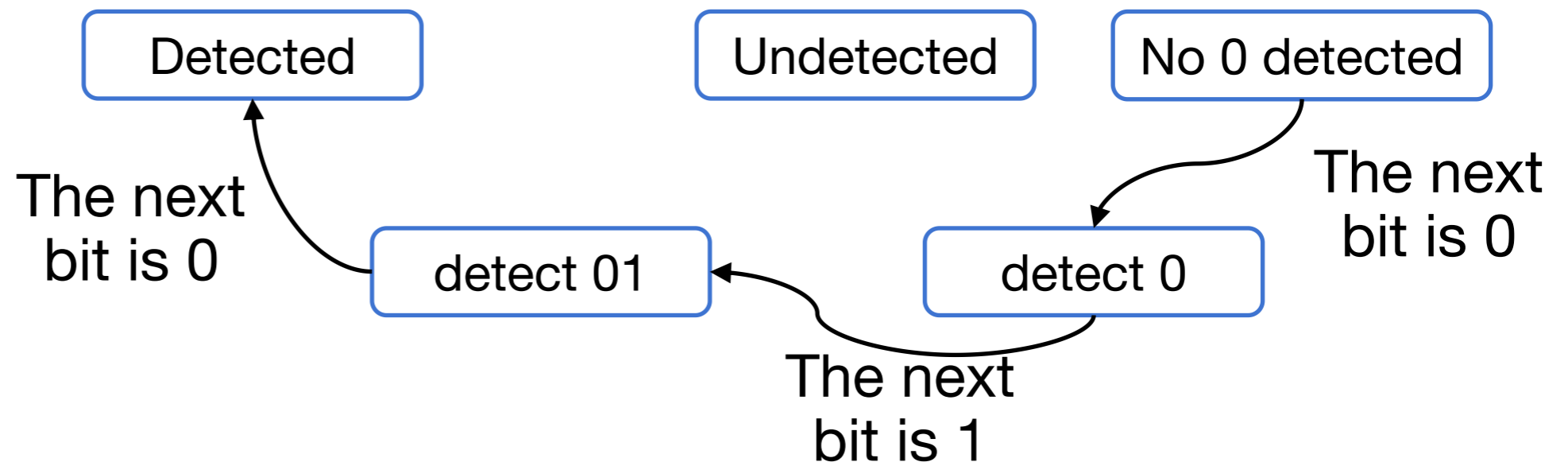


Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function

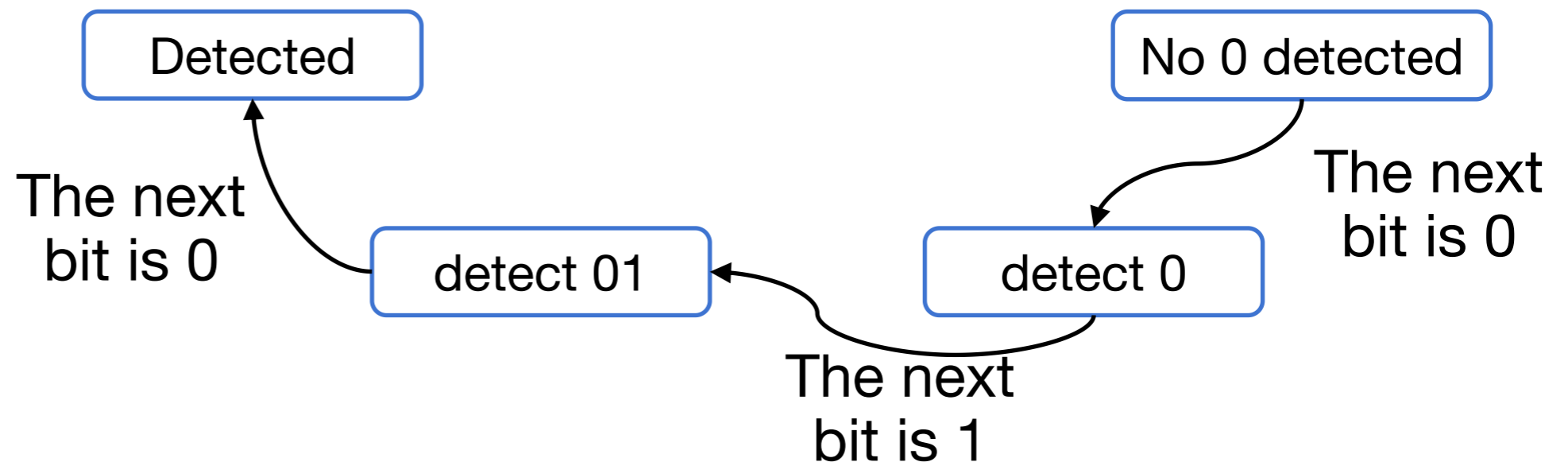


Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function

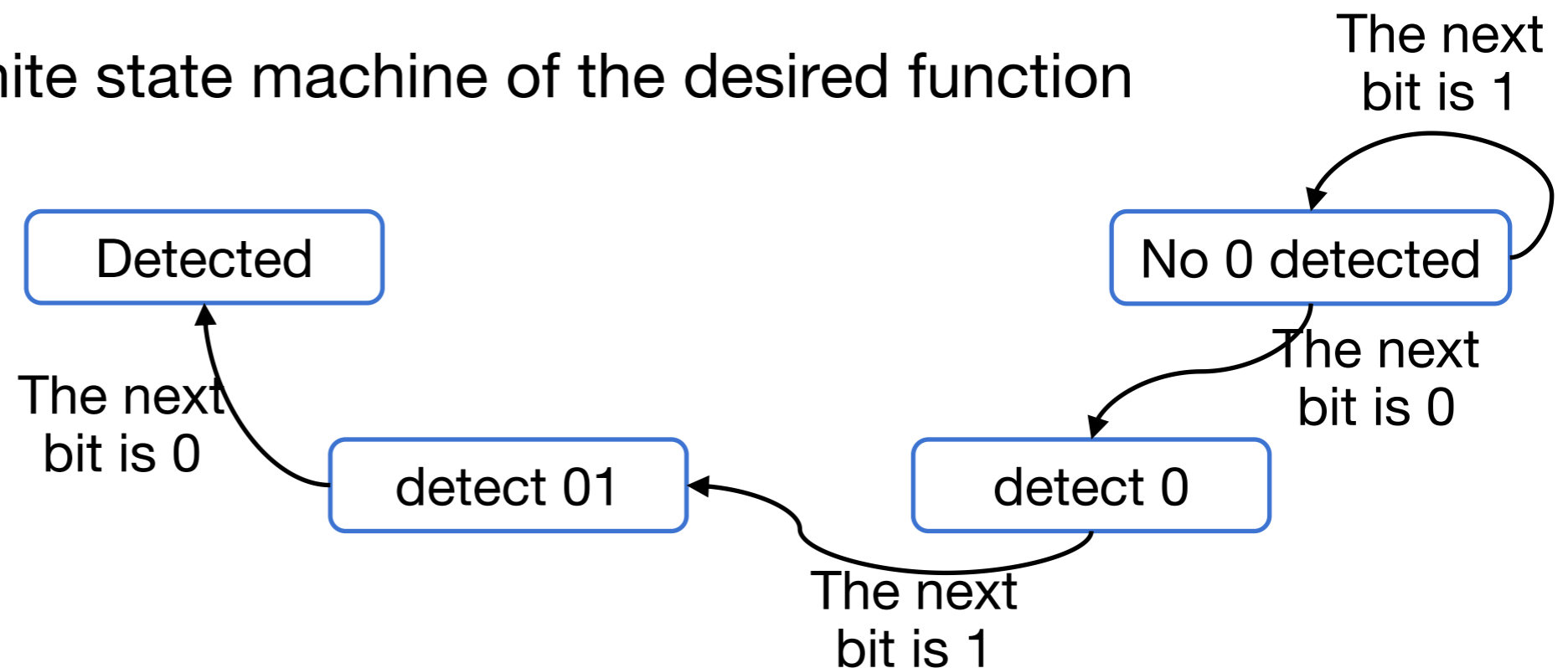


Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function



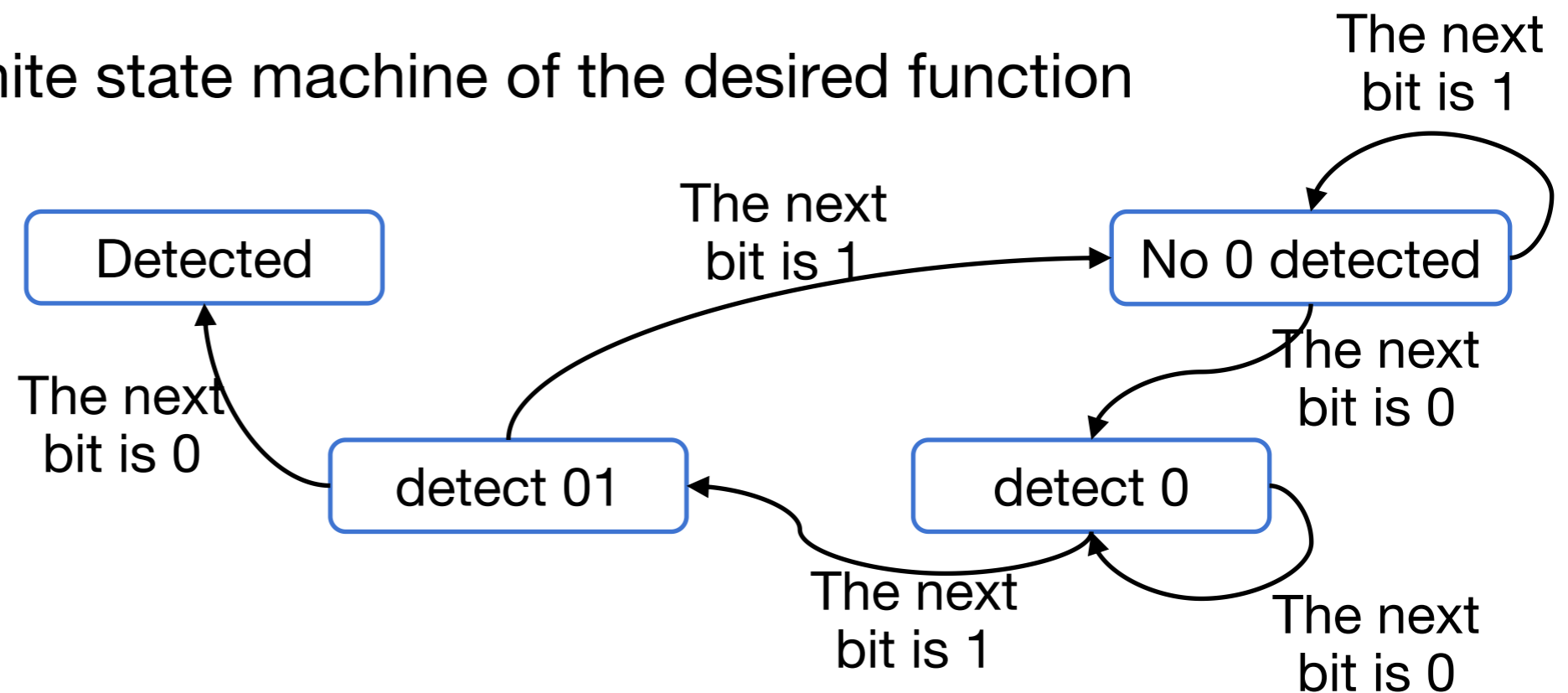
Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function



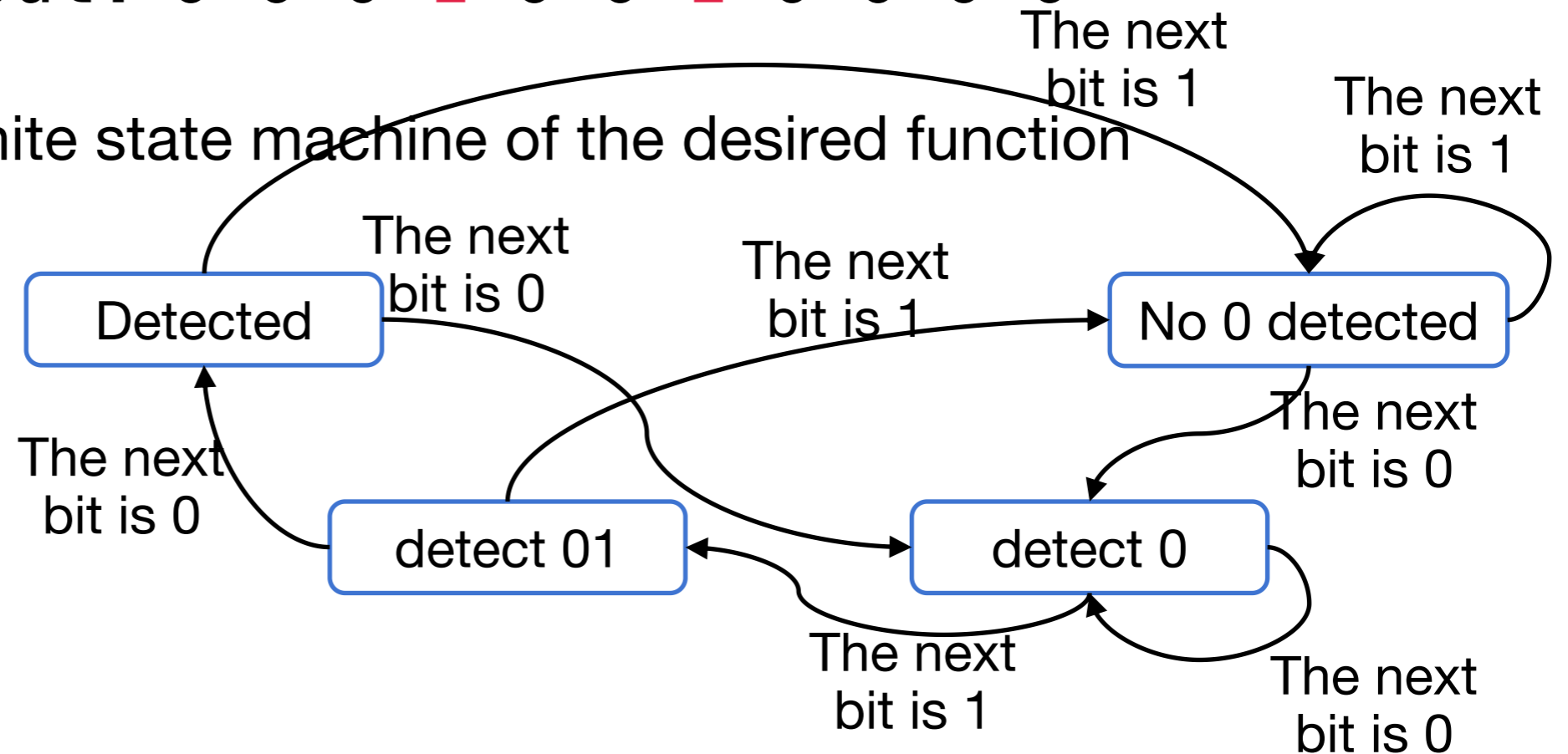
Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function



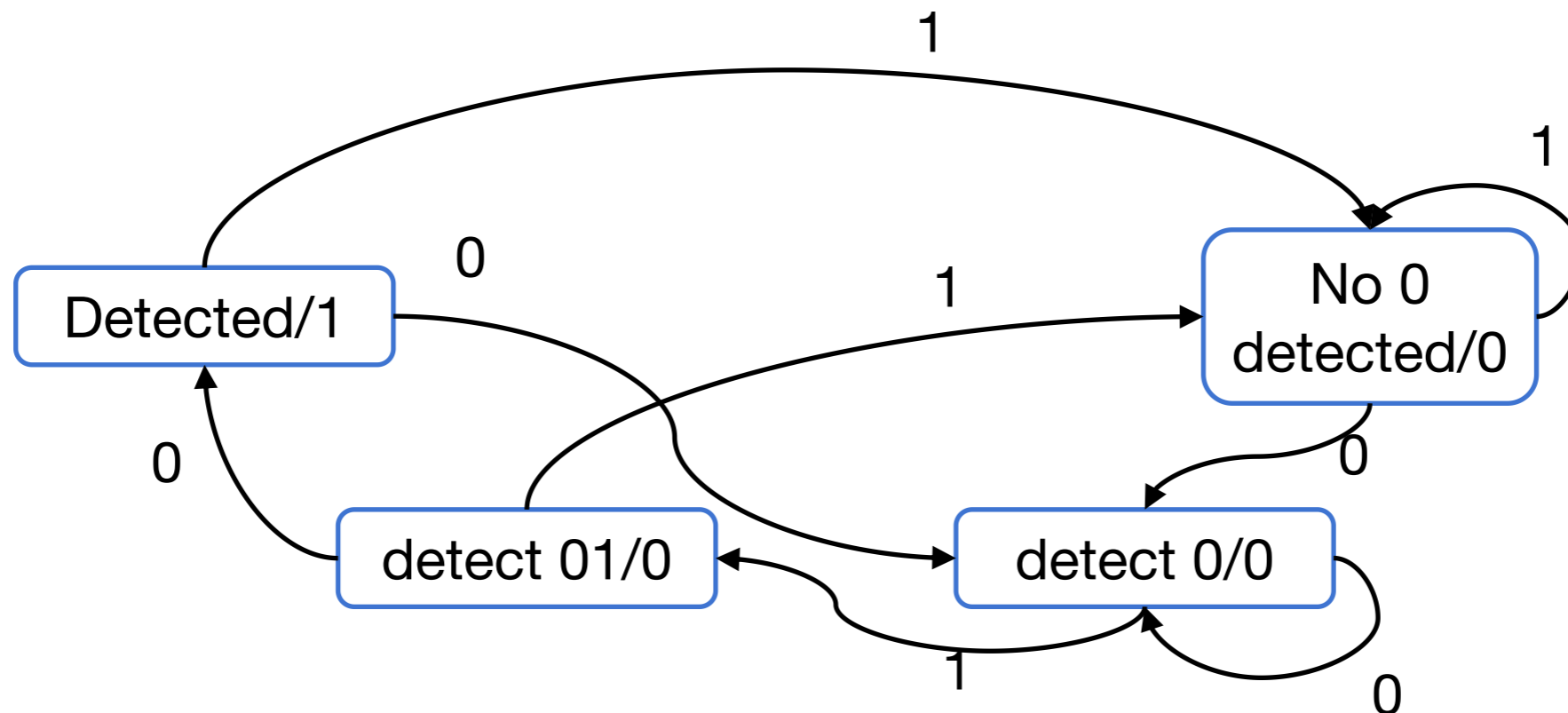
Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function



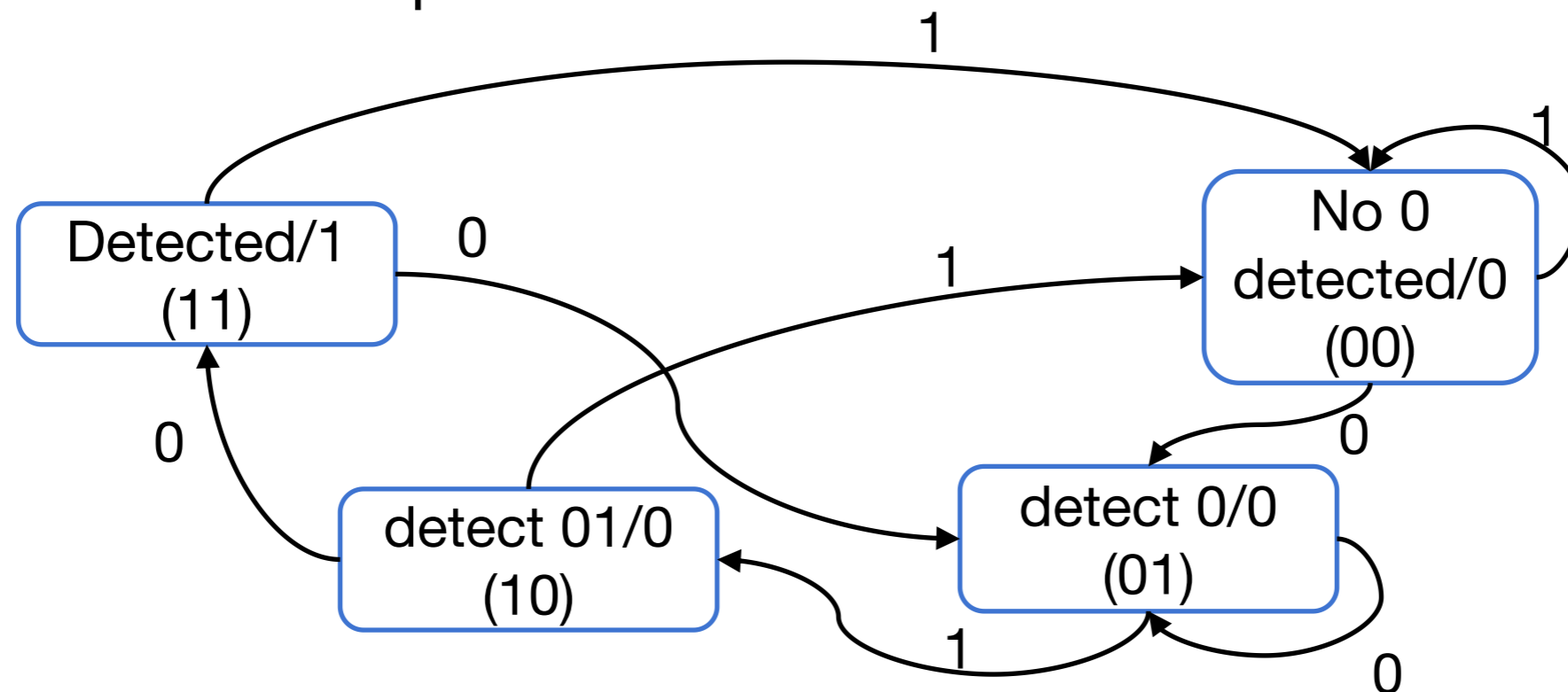
Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 2: Define/assign binary numbers to represent the states, the inputs and the outputs



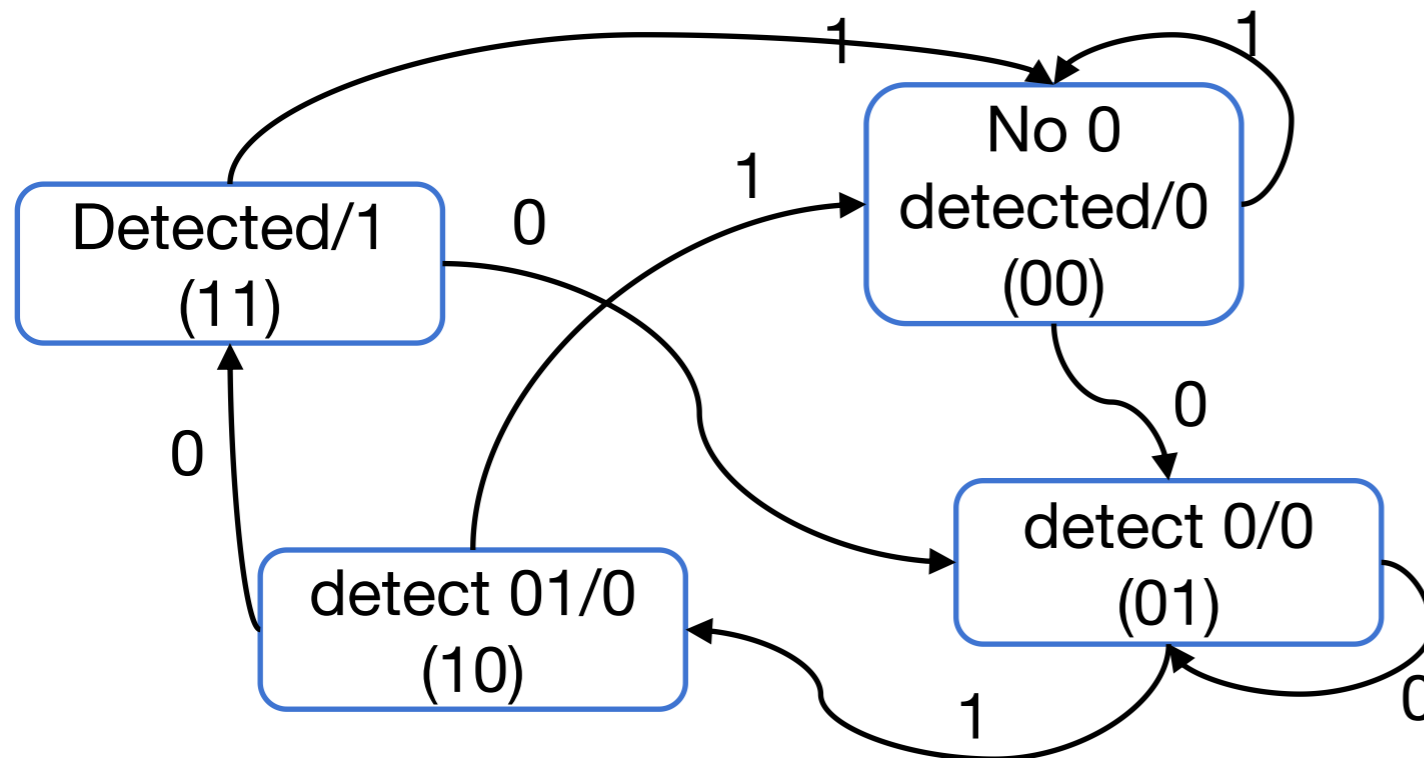
Warm-up

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

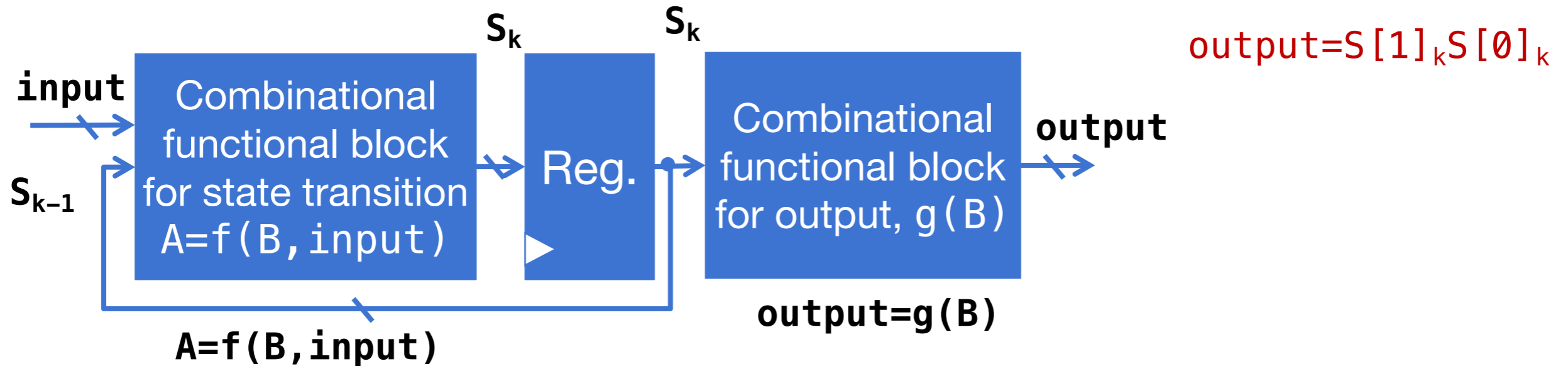
- Step 3: Write down the truth table (enumerate input/previous state (and current state) and their corresponding current state (and output))



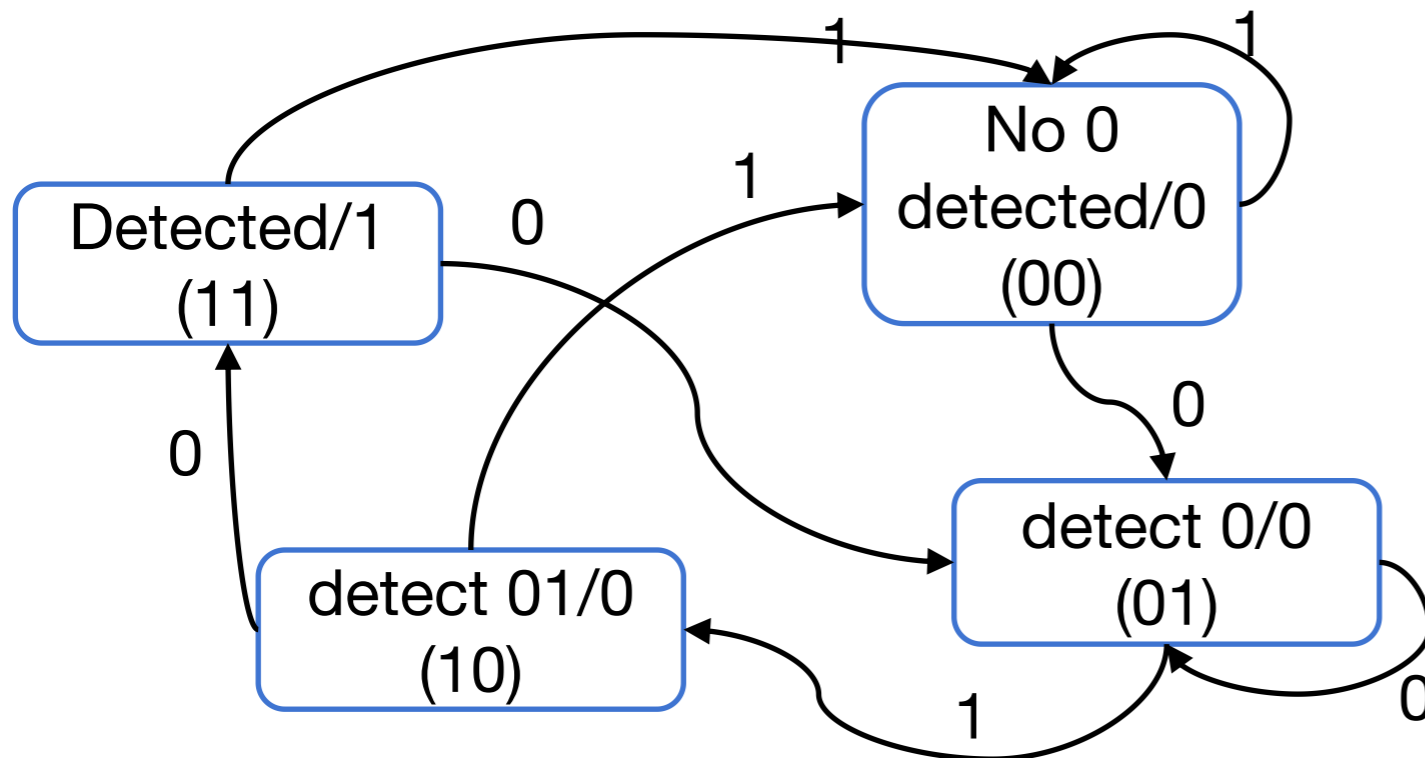
Previous state Current state

input	S[1] k-1	S[0] k-1	S[1] k	S[0] k	output
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Warm-up

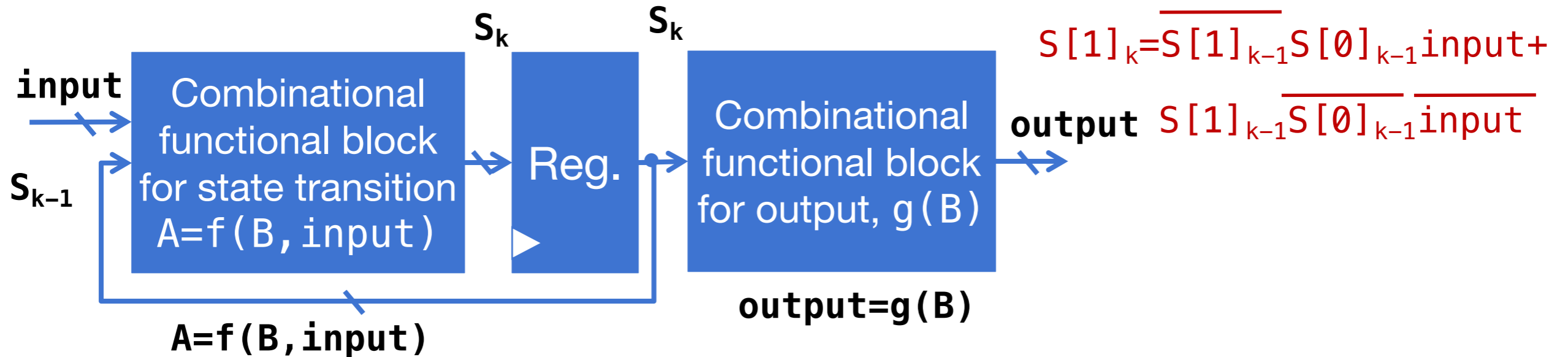


- Step 4: Use template and decide the combinational block for state transition and output logic

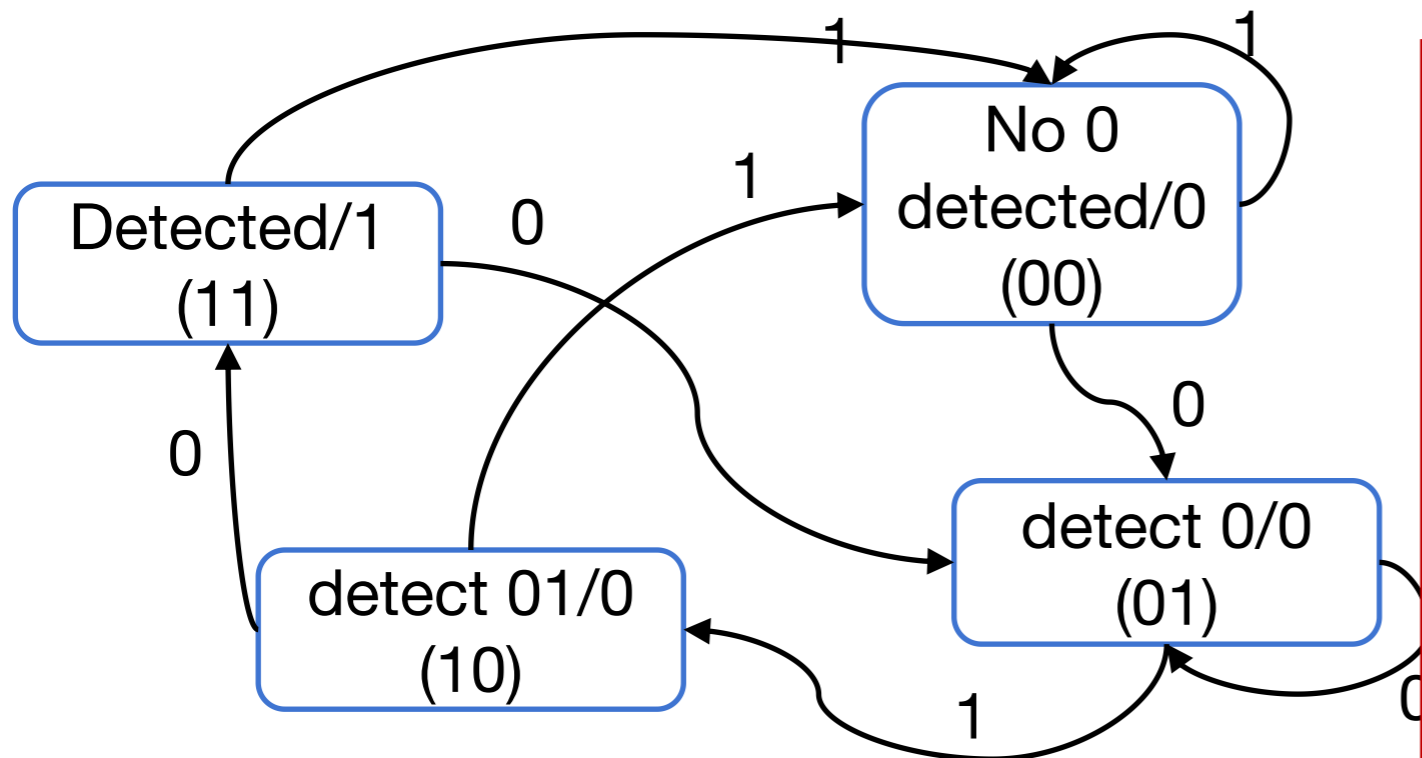


input	Previous state		Current state		output
	$S[1]_{k-1}$	$S[0]_{k-1}$	$S[1]_k$	$S[0]_k$	
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Warm-up

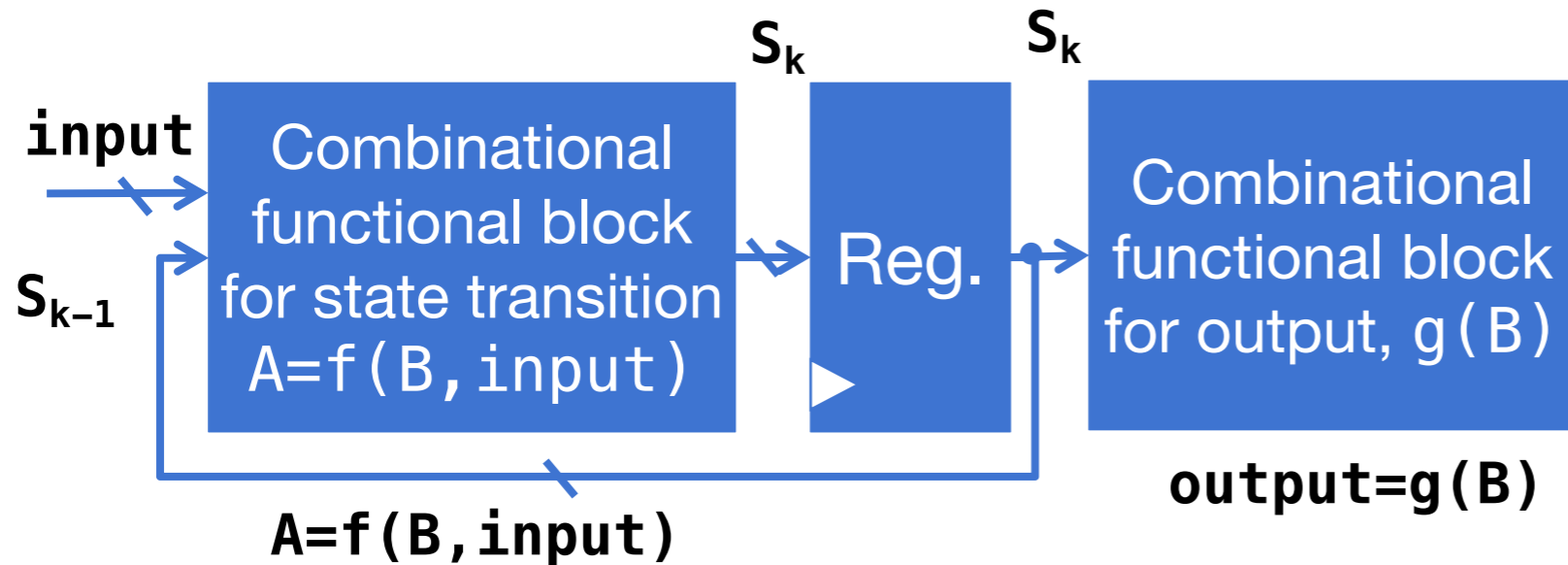


- Step 4: Use template and decide the combinational block for state transition and output logic



	Previous state		Current state		
input	S[1] _{k-1}	S[0] _{k-1}	S[1] _k	S[0] _k	output
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Warm-up

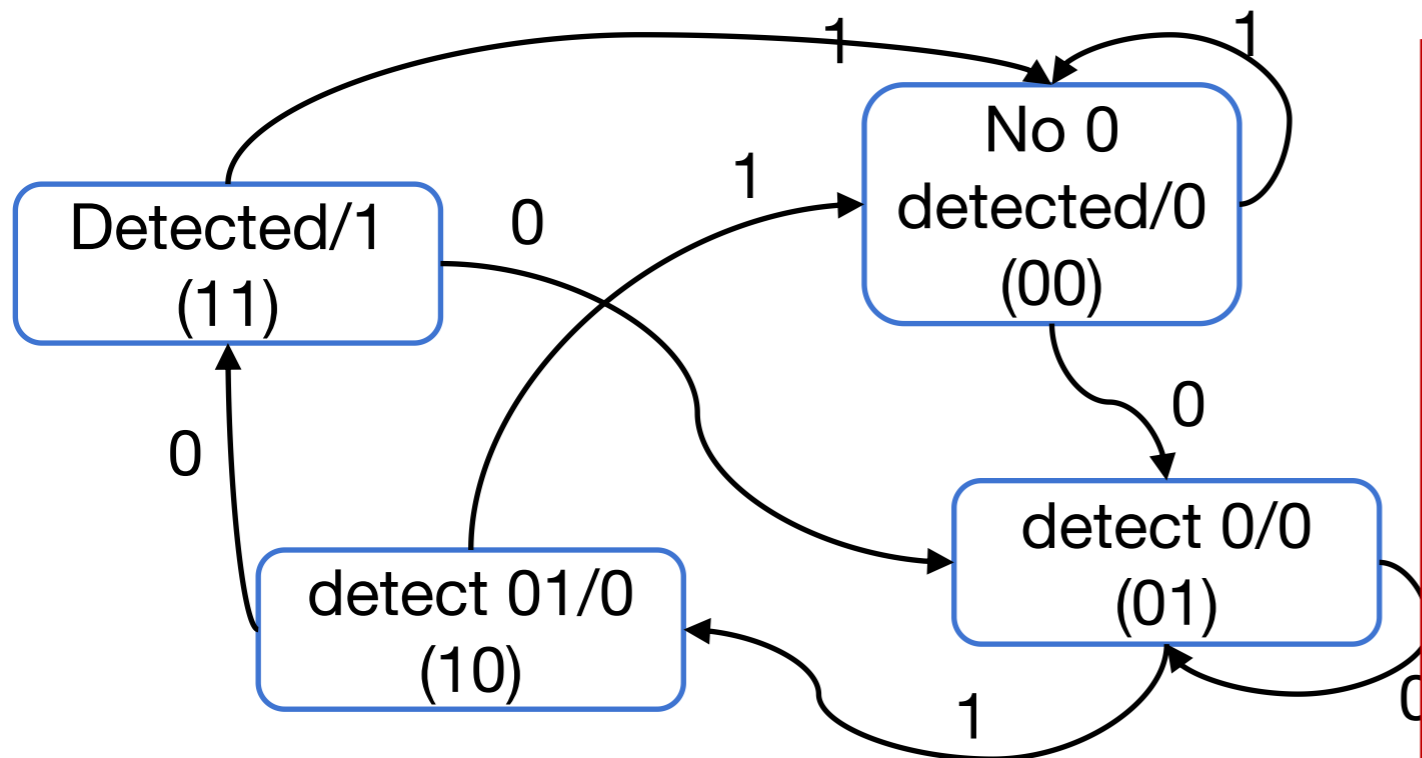


$$\text{output} = S[1]_k S[0]_k$$

$$S[1]_k = \overline{S[1]_{k-1}} S[0]_{k-1} \text{input} + S[1]_{k-1} \overline{S[0]_{k-1}} \text{input}$$

$$S[0]_k = \text{input}$$

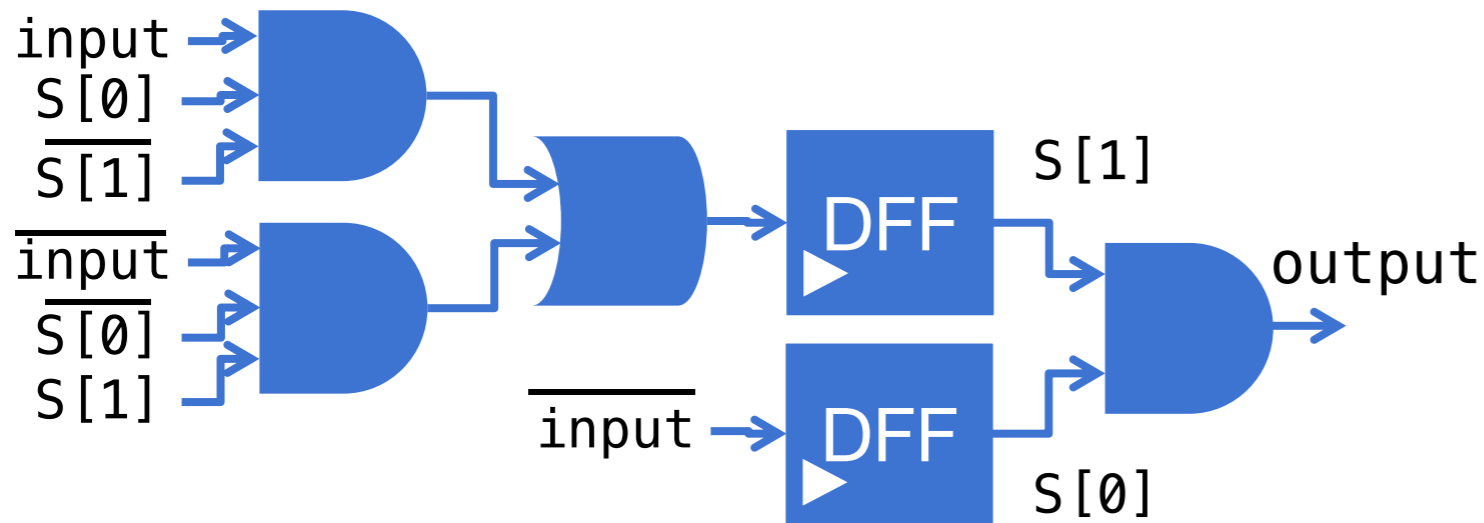
- Step 4: Use template and decide the combinational block for state transition and output logic



Previous state Current state

input	S[1] _{k-1}	S[0] _{k-1}	S[1] _k	S[0] _k	output
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Warm-up

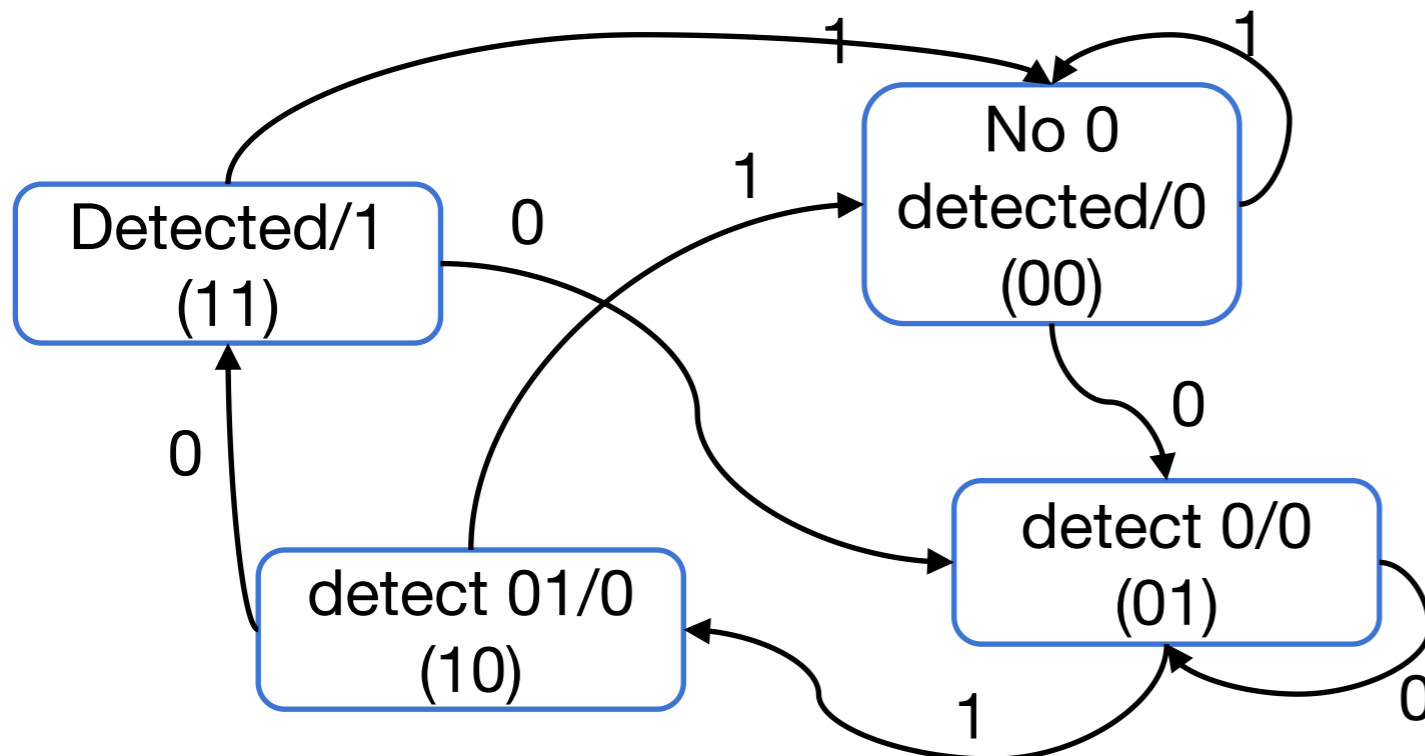


$$\text{output} = S[1]_k S[0]_k$$

$$S[1]_k = \overline{S[1]_{k-1}} S[0]_{k-1} \text{input} + S[1]_{k-1} \overline{S[0]_{k-1}} \text{input}$$

$$S[0]_k = \overline{S[0]_{k-1}} \text{input}$$

- Step 4: Use template and decide the combinational block for state transition and output logic

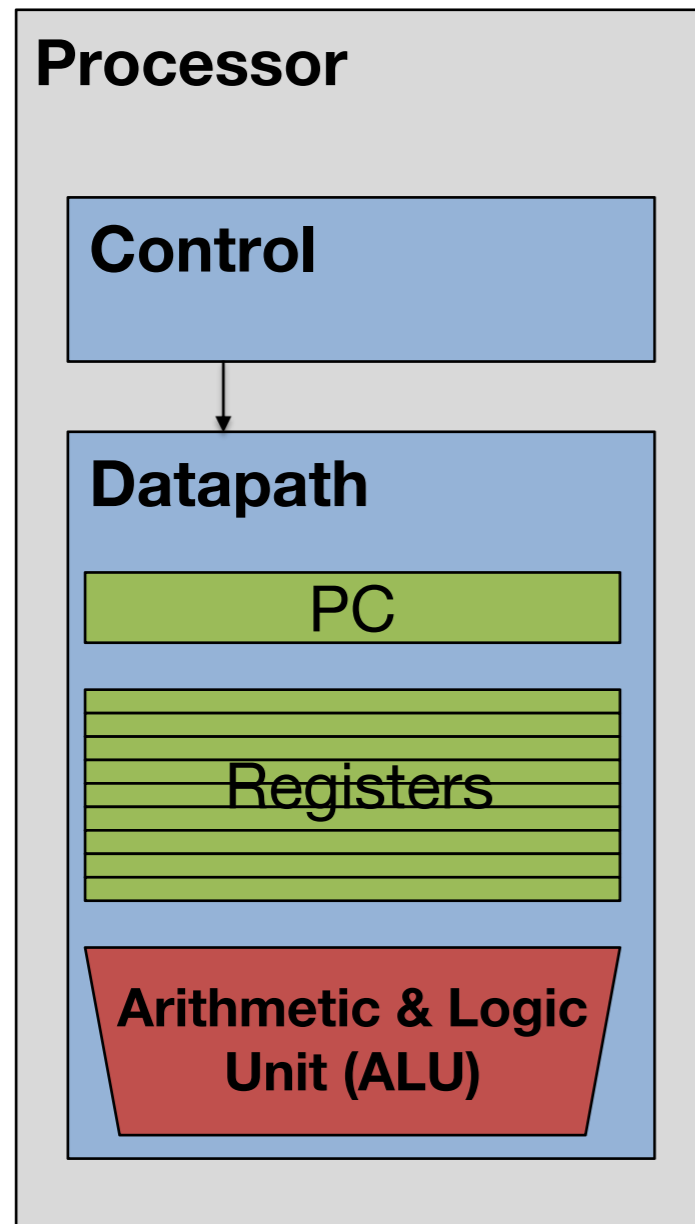


Previous state Current state

input	S[1] _{k-1}	S[0] _{k-1}	S[1] _k	S[0] _k	output
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Controller & Datapath

- A CPU that support RV32I can have so many states

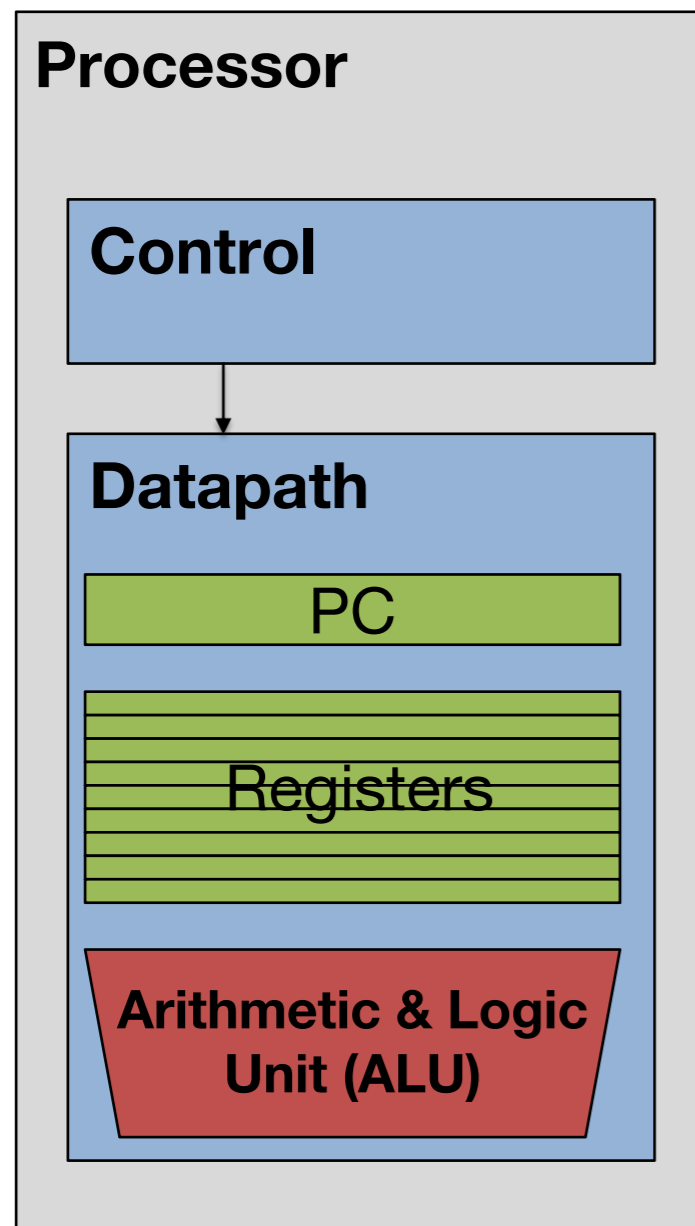


- Consider the 32 registers alone
 - $x0$ always 0
 - Each bit in the other registers can be 0 or 1
- Not practical to enumerate all the state transitions
- Top-down design: build small modules and then connect them as needed
- Most digital systems can be divided into datapath and controller
 - Datapath contains data processing and storage
 - Controller controls data flow and state change (still can be modeled as FSM)
- Recall the execution of an instruction

- Our Goal: Implement a RISC-V processor as a synchronous digital system.
- Each RV32I instruction can be done within 1 clock cycle (single-cycle CPU).

Controller & Datapath

- A CPU that support RV32I can have so many states

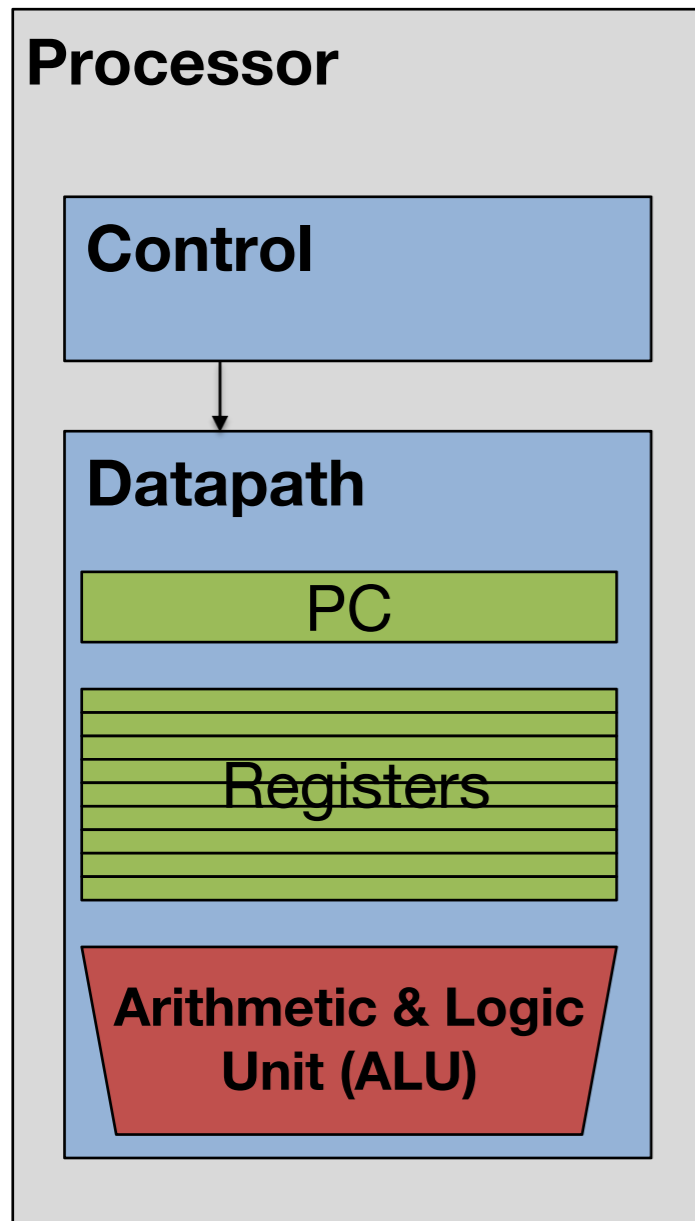


- Datapath
 - Start with basic building blocks
 - Add building blocks to the digital system with added supported instructions
- Controller
 - Can be considered as an FSM

- Our Goal: Implement a RISC-V processor as a synchronous digital system.
- Each RV32I instruction can be done within 1 clock cycle (single-cycle CPU).

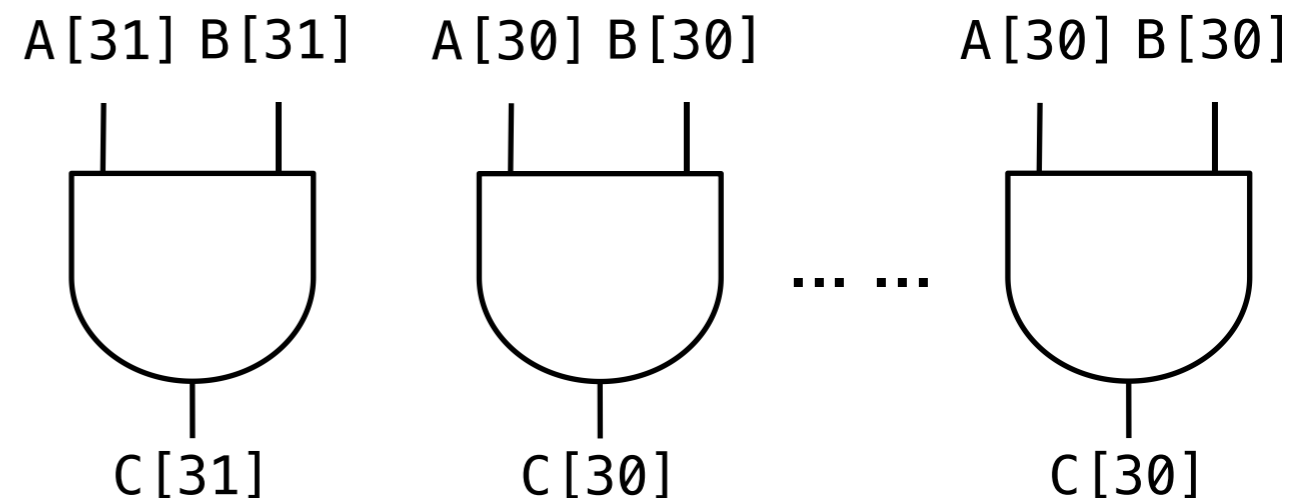
Useful building blocks

- An ALU should be able to execute all the arithmetic and logic operations



ADD	ADDI
SUB	SLTI
SLL	SLTIU
SLT	XORI
SLTU	ORI
XOR	ANDI
SRL	
SRA	
OR	
AND	

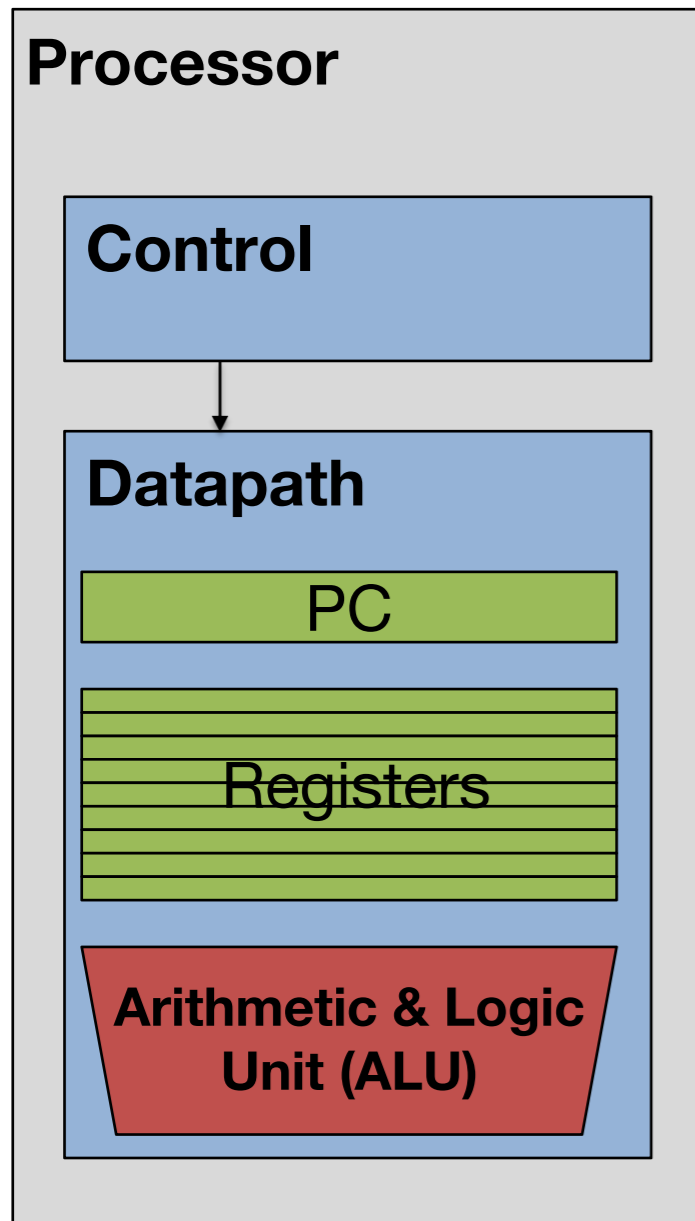
- AND as an example
 - 2 32-bit inputs A and B
 - 1 32-bit output C



- Our Goal: Implement a RISC-V processor as a synchronous digital system.
- Each RV32I instruction can be done within 1 clock cycle.

Useful building blocks

- An ALU should be able to execute all the arithmetic and logic operations



ADD

SUB

SLL

SLT

SLTU

XOR

SRL

SRA

OR

AND

ADDI

SLTI

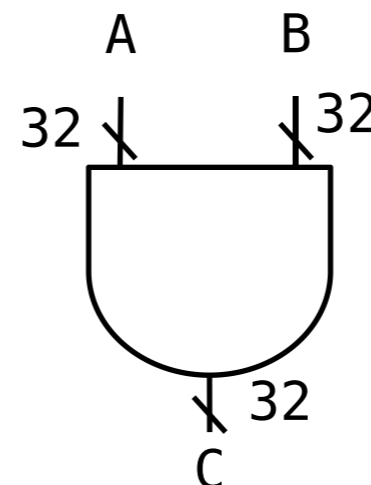
SLTIU

XORI

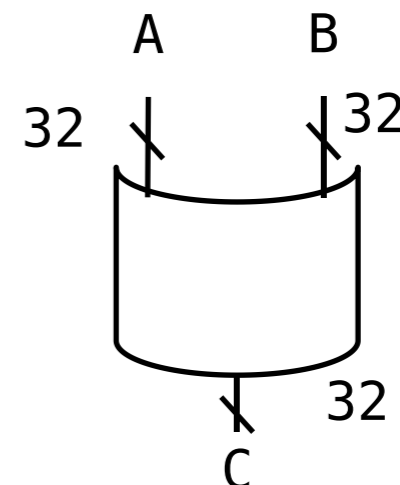
ORI

ANDI

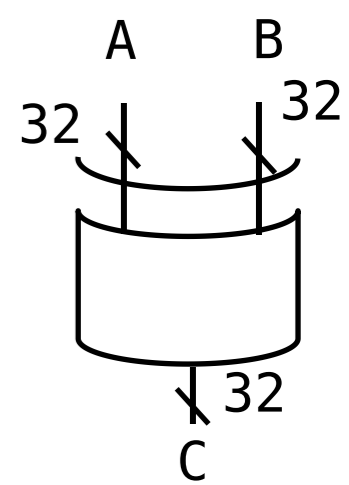
- AND as an example
 - 2 32-bit inputs A and B
 - 1 32-bit output C



A simplified AND gate array symbol



OR

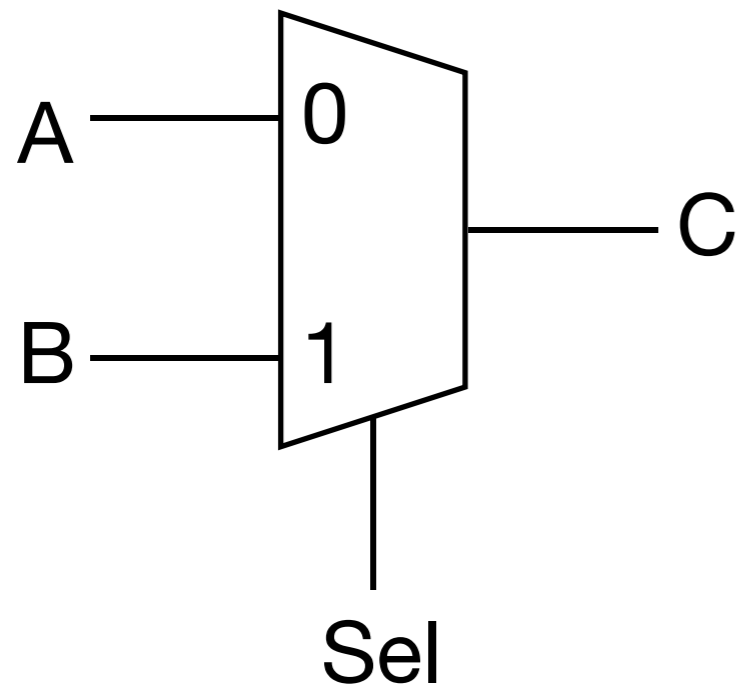


XOR

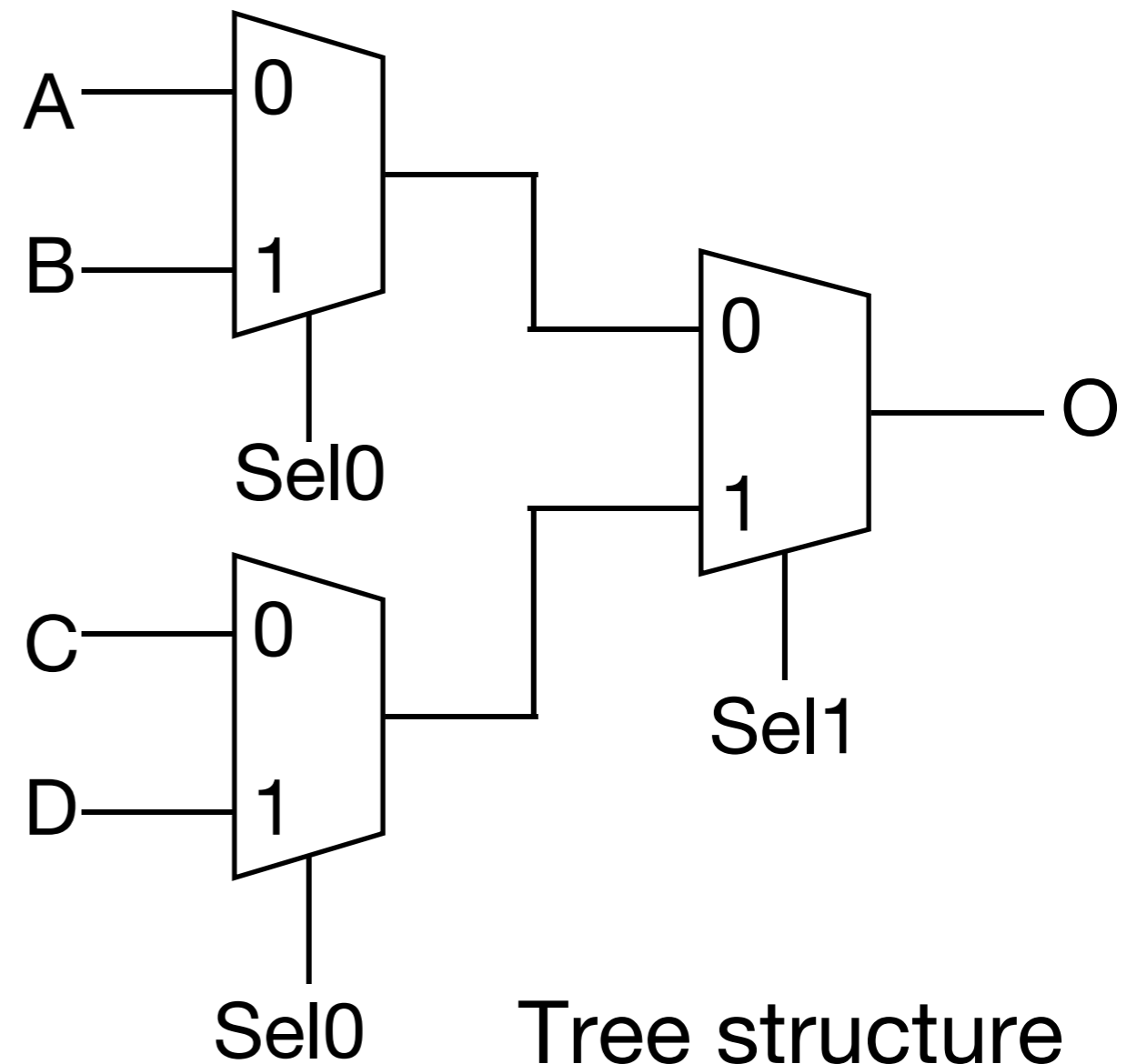
- Our Goal: Implement a RISC-V processor as a synchronous digital system.
- Each RV32I instruction can be done within 1 clock cycle.

Useful Combinational Circuits

- Multiplexer (2-to-1)

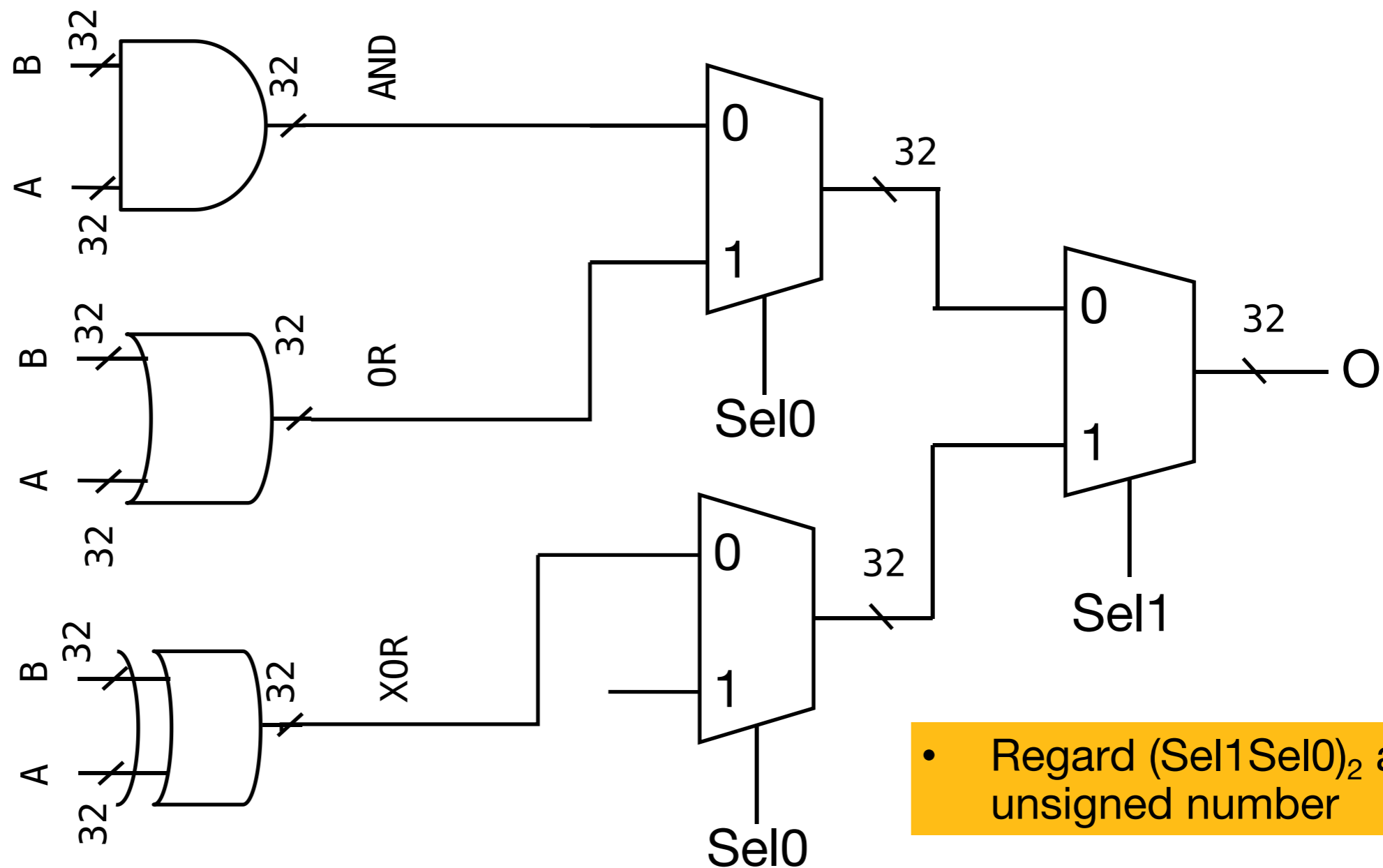


- Multiplexer (2^n -to-1)



Control through selection

- 32-bit Multiplexer and logic gates to support some logic instructions

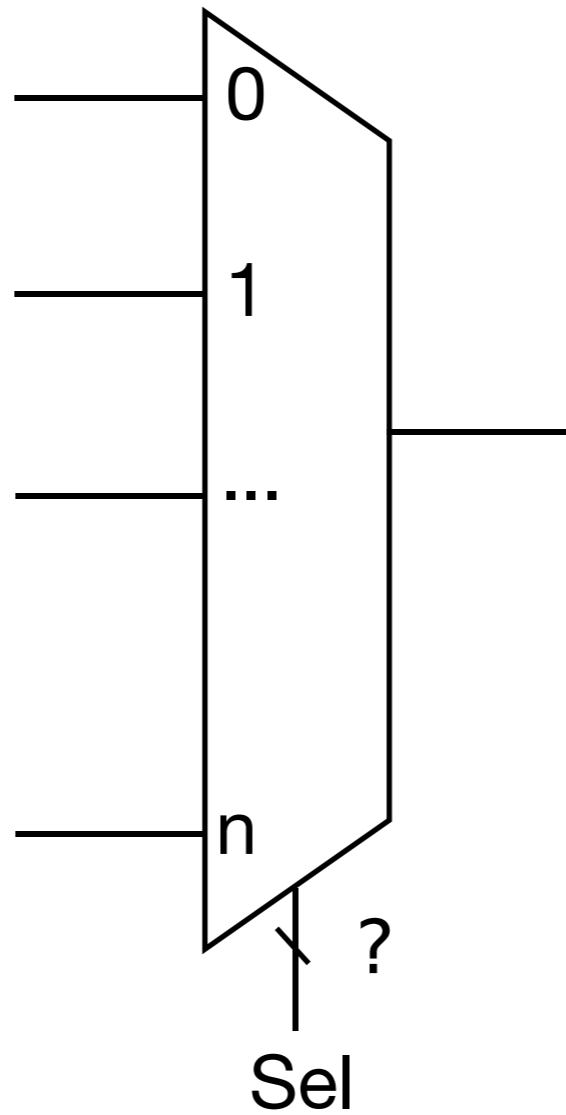


- Regard $(Sel1Sel0)_2$ as an unsigned number

- More layers of multiplexer to select from more inputs

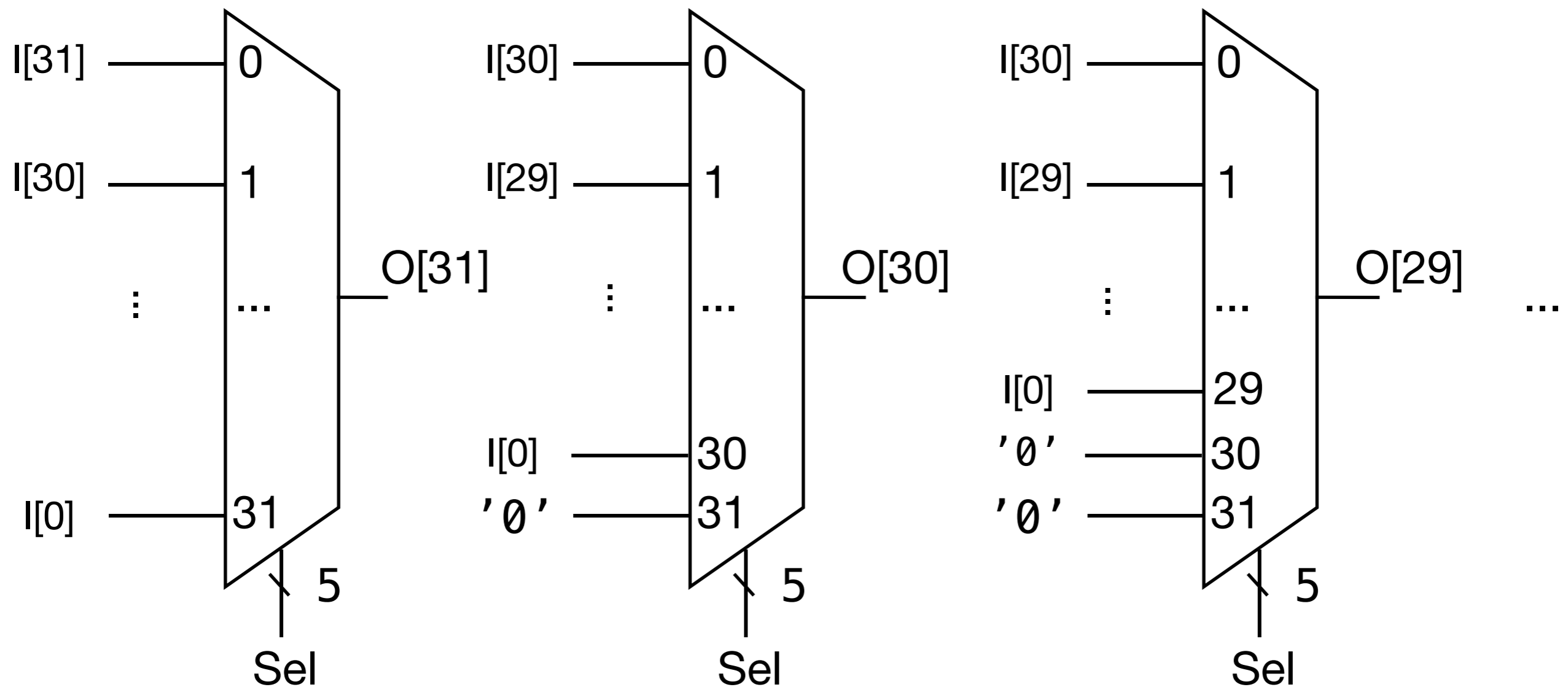
Multiplexer

- n-to-1 multiplexer symbol



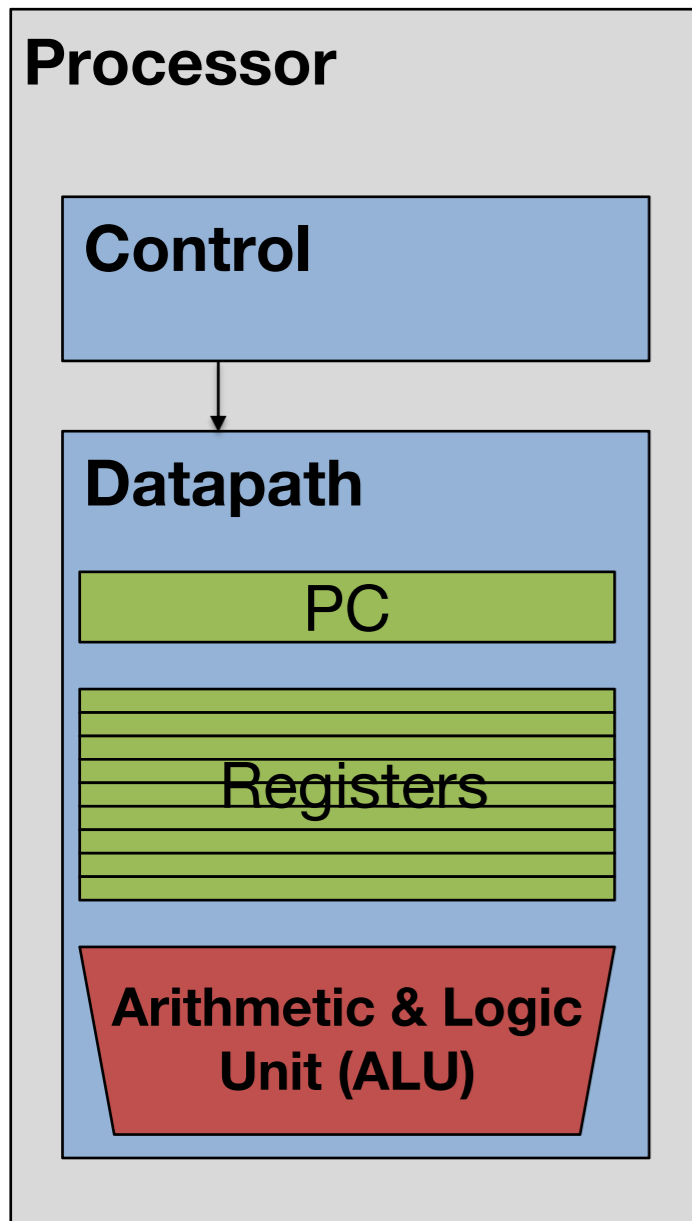
Multiplexers used for shifter

- Left shift a single bit -> left shift multiple single bits
- Other shifter designs such as barrel shifter

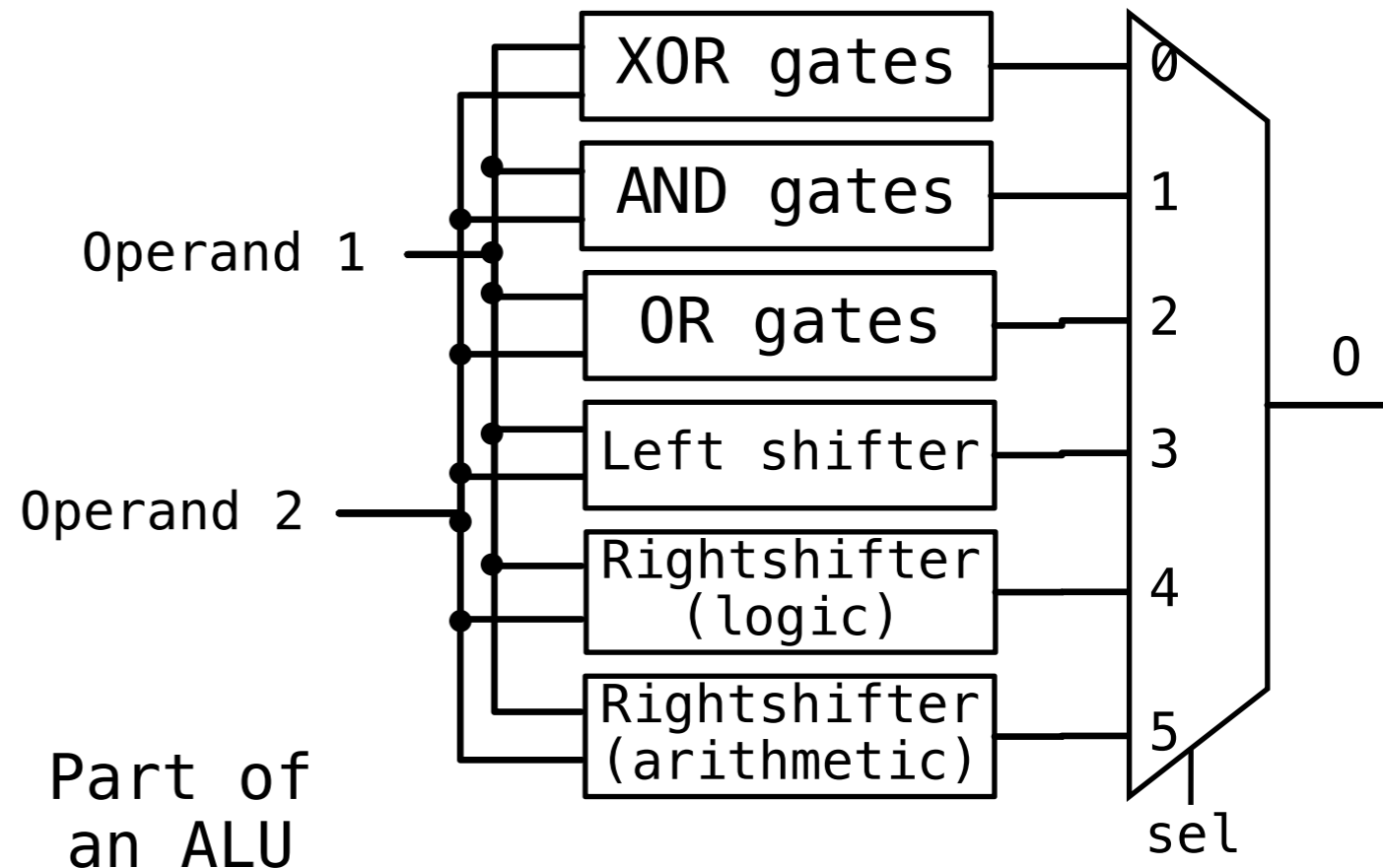


Useful building blocks

- An ALU should be able to execute all the arithmetic and logic operations



ADD	ADDI
SUB	SLTI
SLL	SLTIU
SLT	XORI
SLTU	ORI
XOR	ANDI
SRL	
SRA	
OR	
AND	

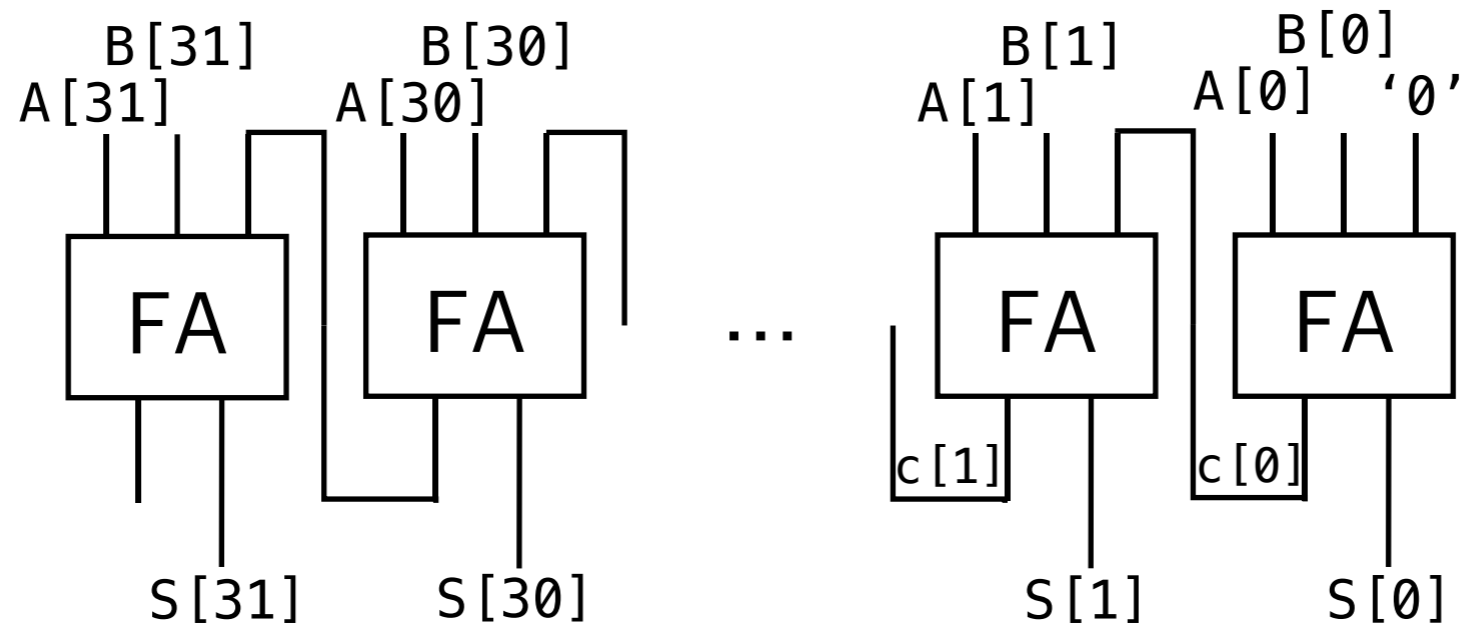


Note that all the signals except the selection signals are 32-bit.

- Our Goal: Implement a RISC-V processor as a synchronous digital system.
- Each RV32I instruction can be done within 1 clock cycle.

Adder & subtractor

- An adder design

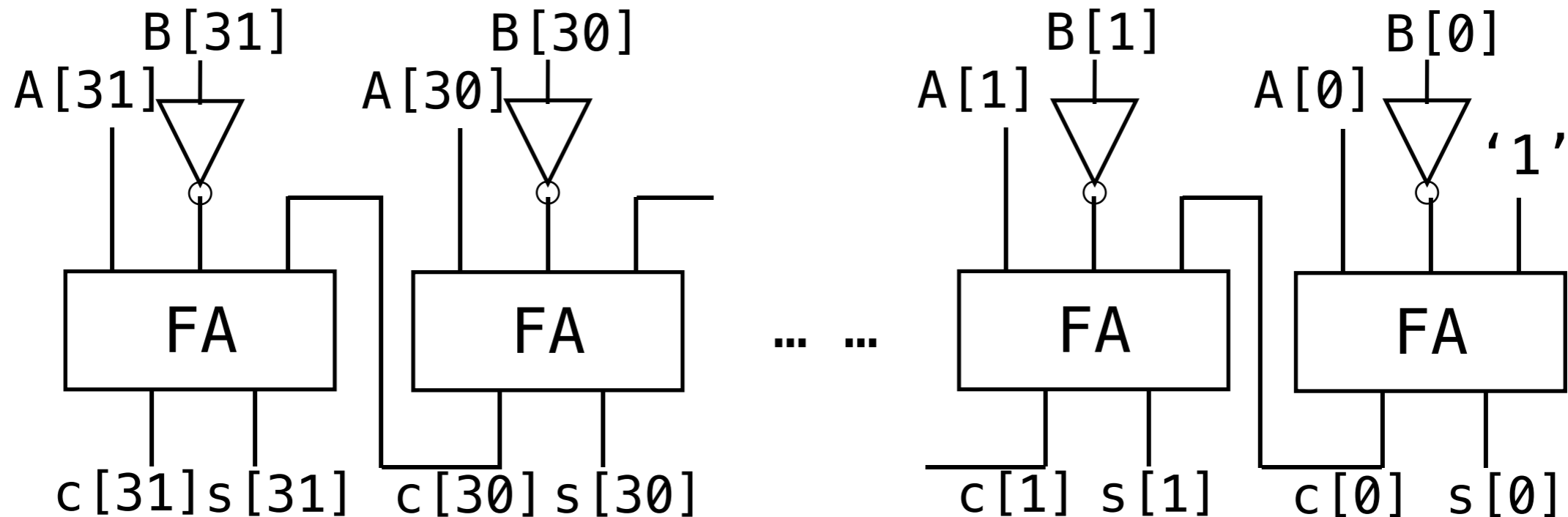


A 32-bit adder

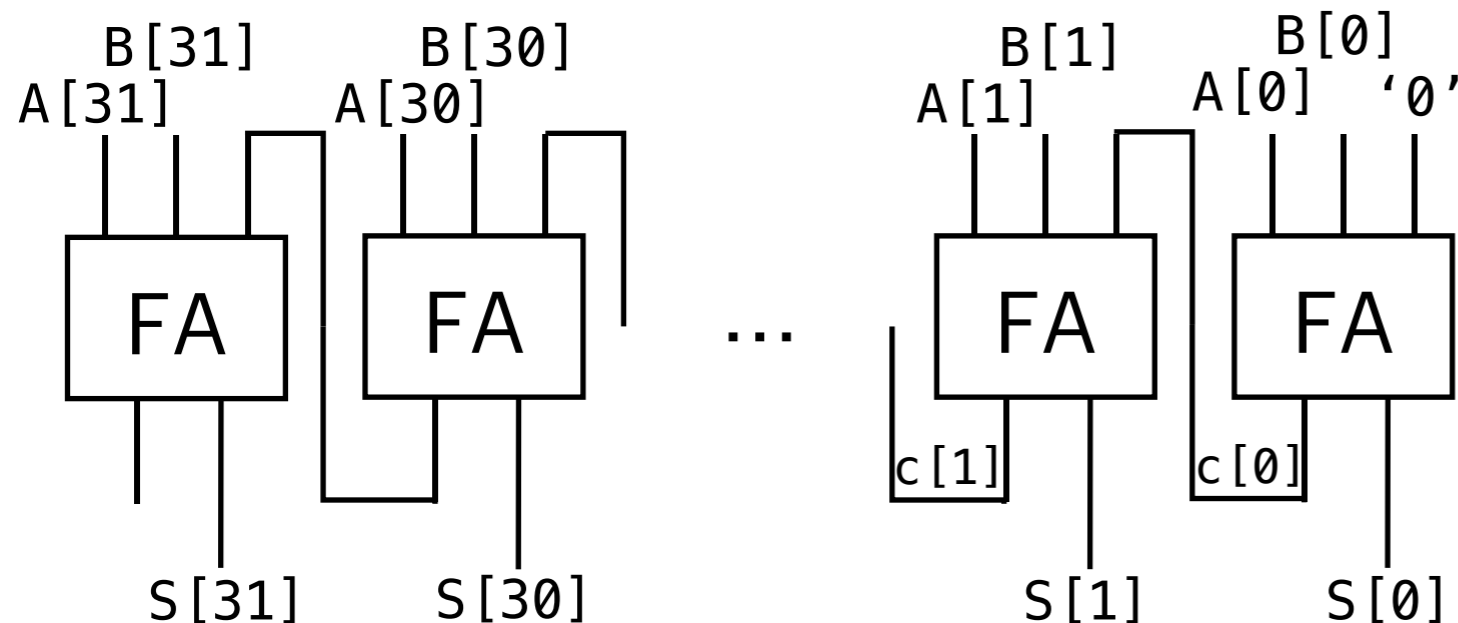
- A smart subtractor design
 - Recall that subtracting a number is equivalent to adding its negative version

A smart subtractor design

$$A - B = A + (-B) = A + \bar{B} + 1 \pmod{2^{N-1}}$$

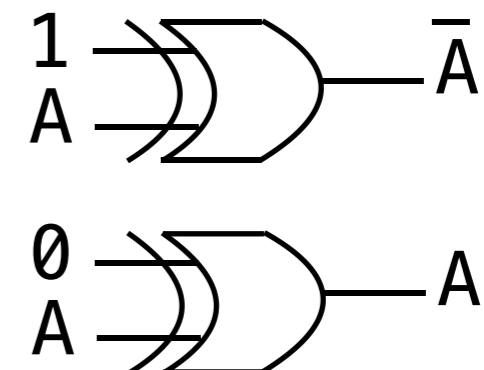


A 32-bit subtractor



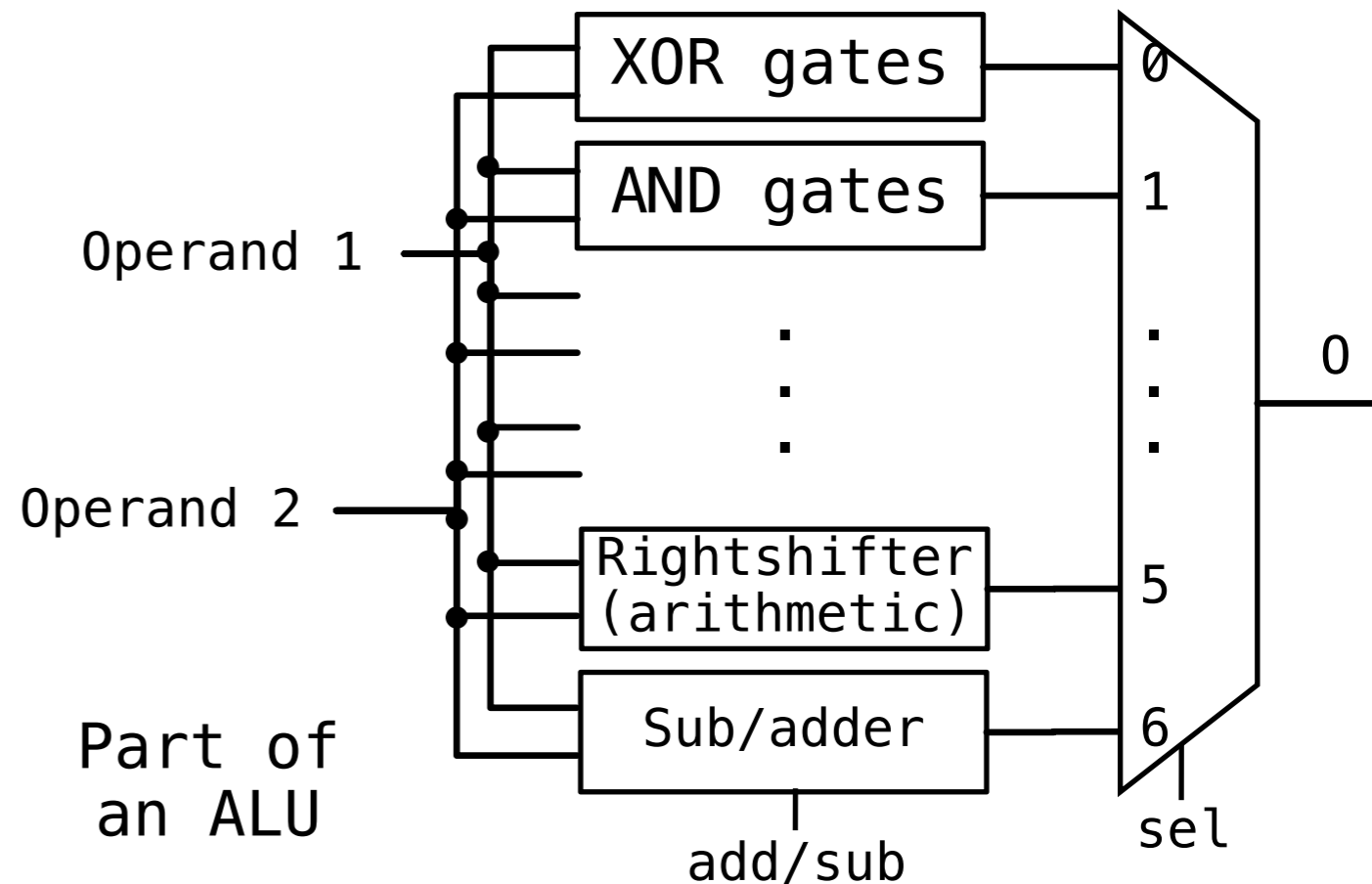
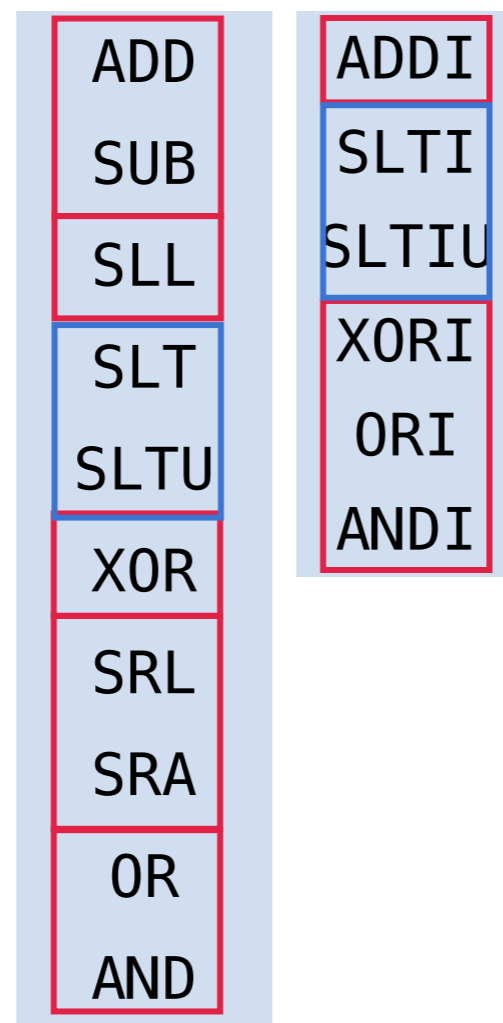
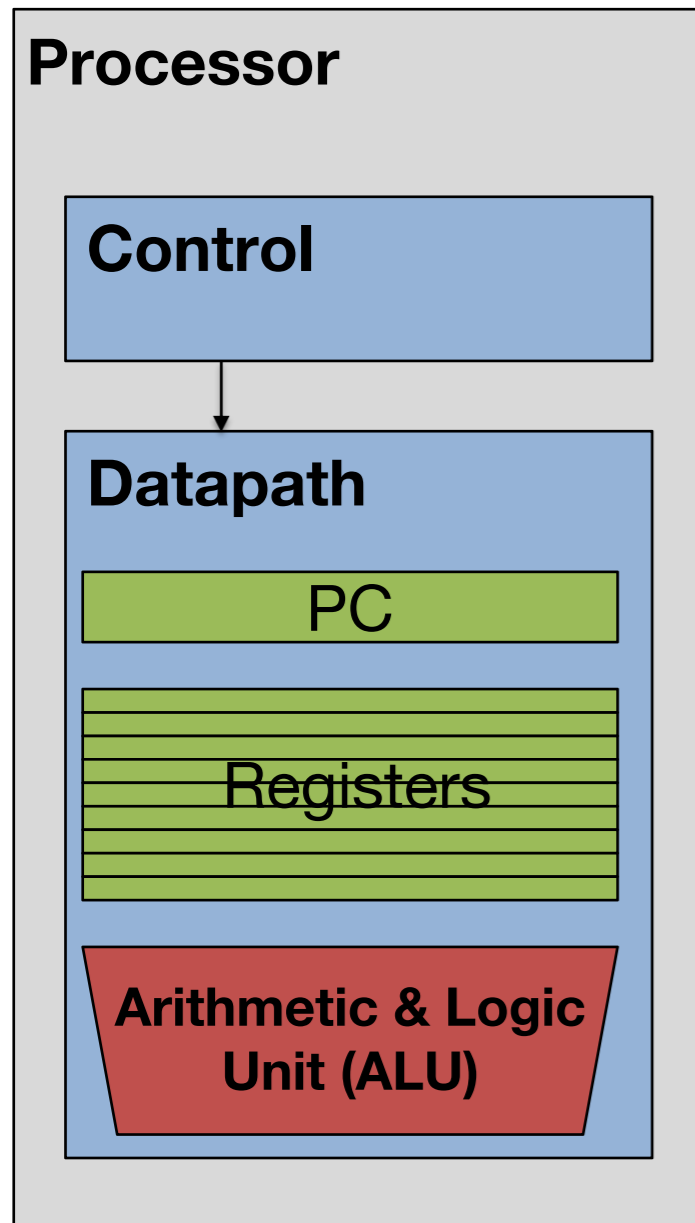
A 32-bit adder

- Recall XOR gate



Useful building blocks

- An ALU should be able to execute all the arithmetic and logic operations

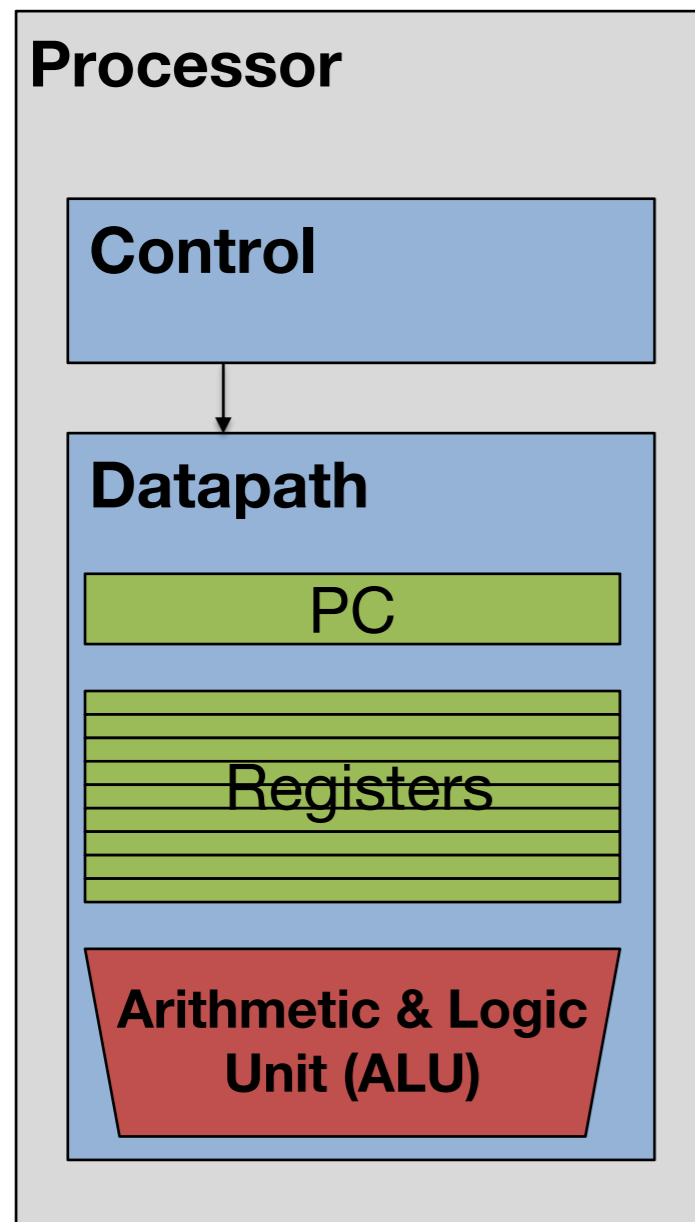


Note that all the signals except the selection signals are 32-bit.

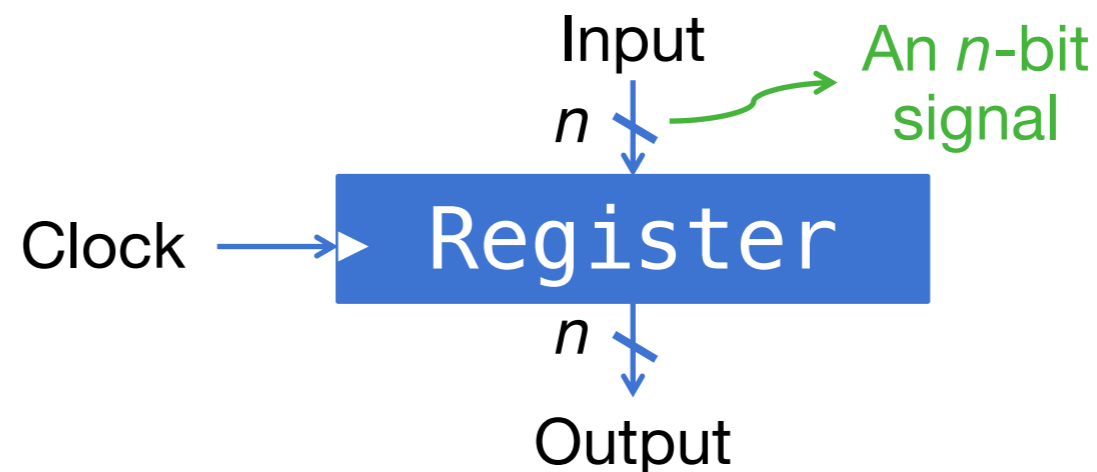
- ALU design that supports R-/I-arithmetic and logic operations completed

Useful building blocks-Register file

- The register file is the component that contains all the general purpose registers of the microprocessor
- A register file should provide data given the register numbers
- A register file should be able to change the stored value

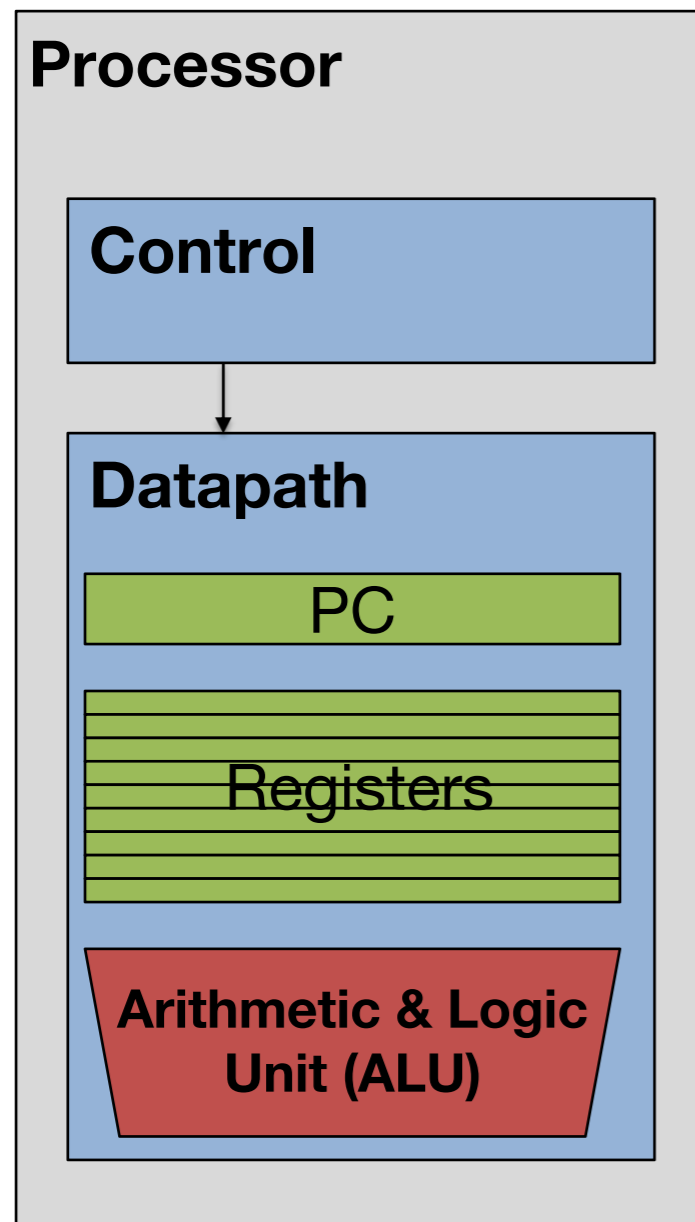


- Recall we have registers that store values

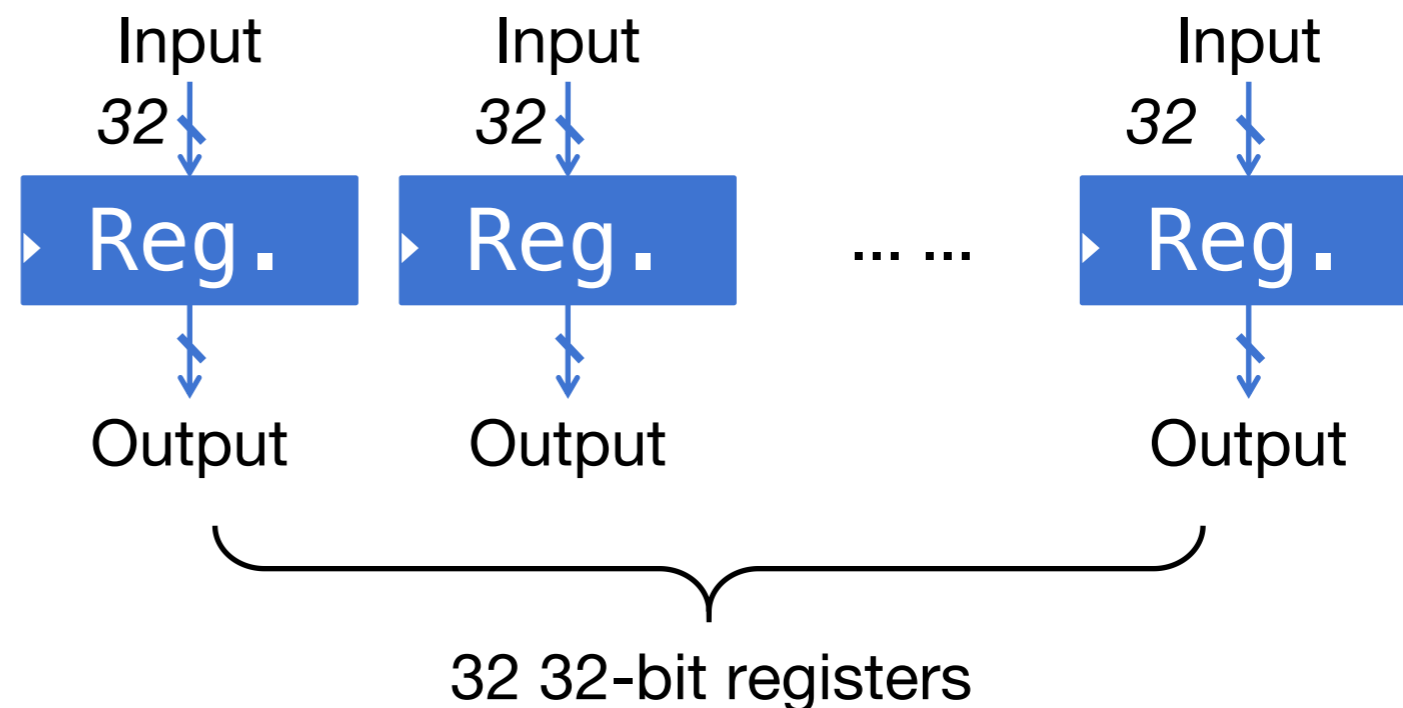


Useful building blocks-Register file

- The register file is the component that contains all the general purpose registers of the microprocessor
- A register file should provide data given the register numbers
- A register file should be able to change the stored value



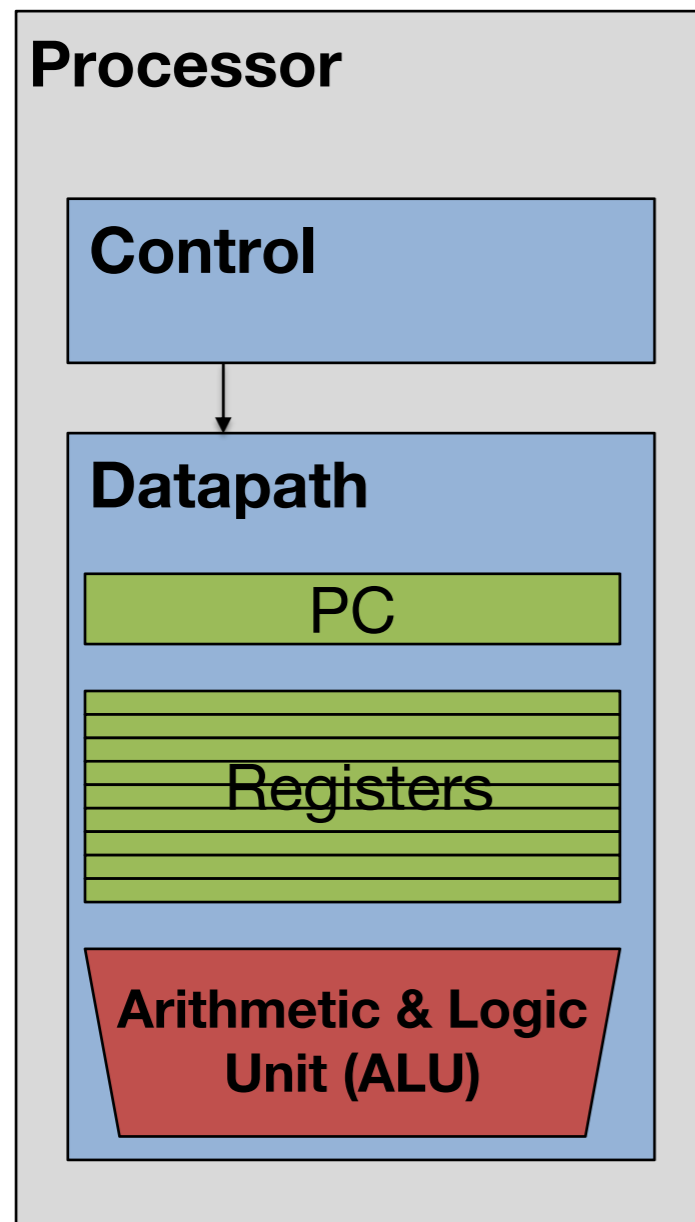
- Recall we have registers that store values



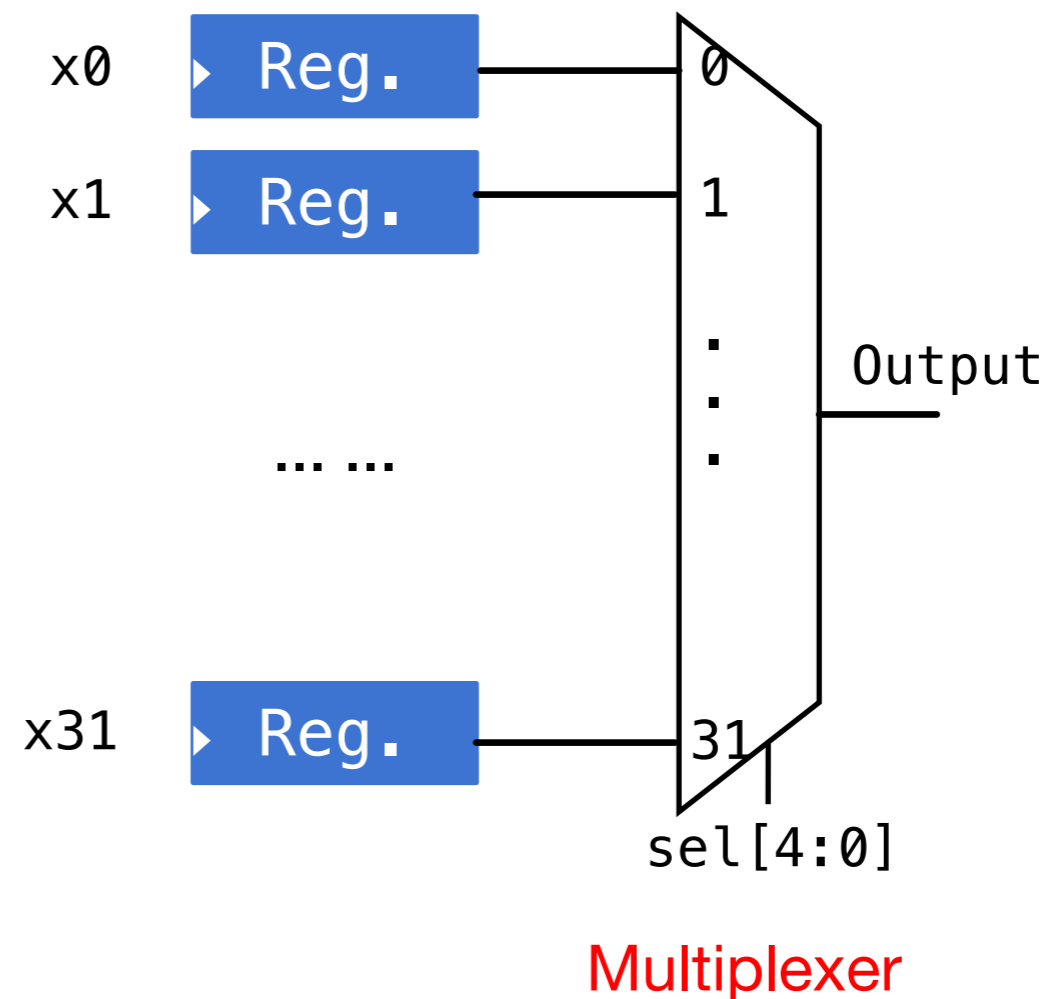
- How to select one to output? **Multiplexer**

Useful building blocks-Register file

- The register file is the component that contains all the general purpose registers of the microprocessor
- A register file should provide data given the register numbers
- A register file should be able to change the stored value

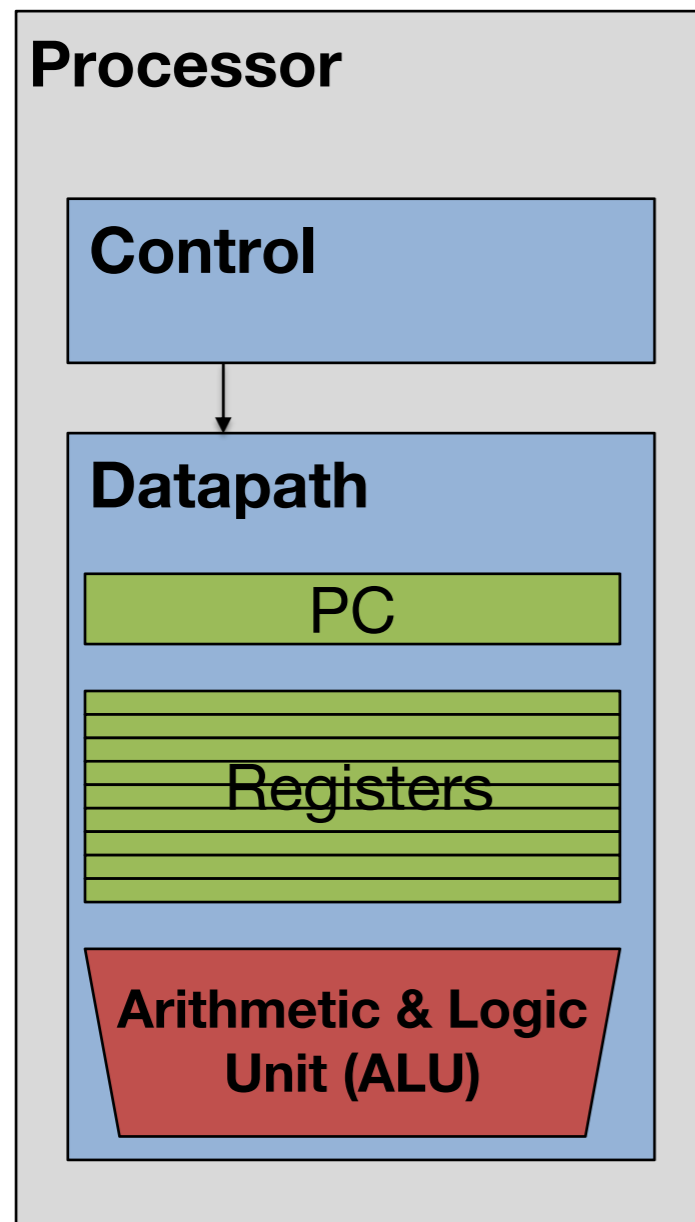


- Recall we have registers that store values

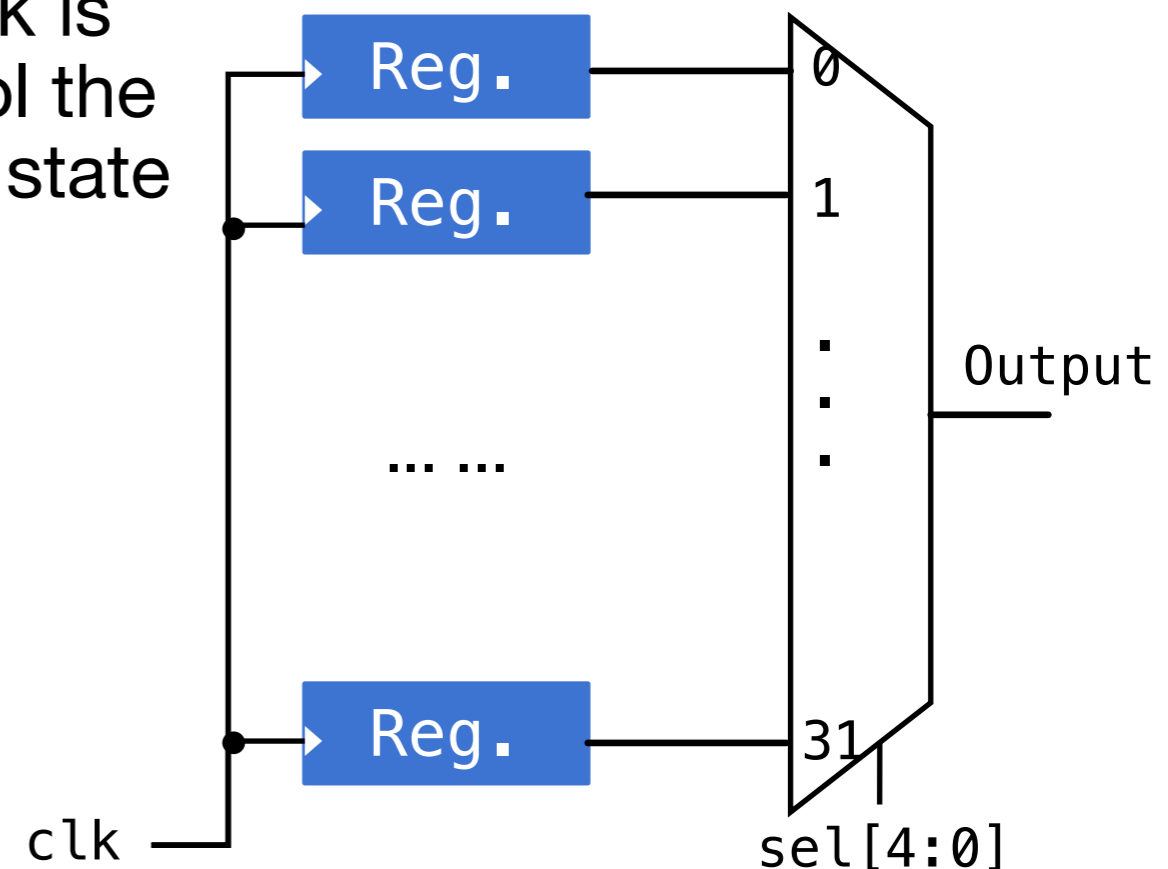


Useful building blocks-Register file

- The register file is the component that contains all the general purpose registers of the microprocessor
- A register file should provide data given the register numbers
- A register file should be able to change the stored value



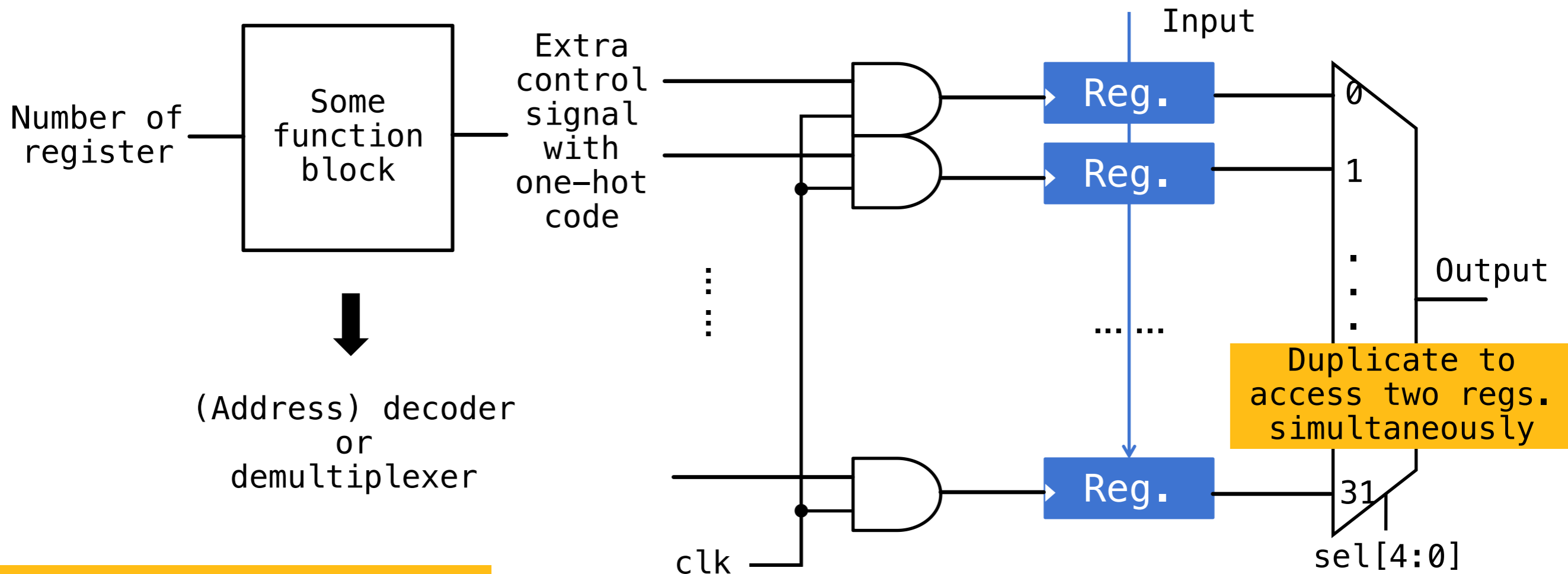
- How do we change values of a specific reg.?
- Recall that `clk` is used to control the change of the state



Useful building blocks-Register file

- The register file is the component that contains all the general purpose registers of the microprocessor
- A register file should provide data given the register numbers
- A register file should be able to change the stored value

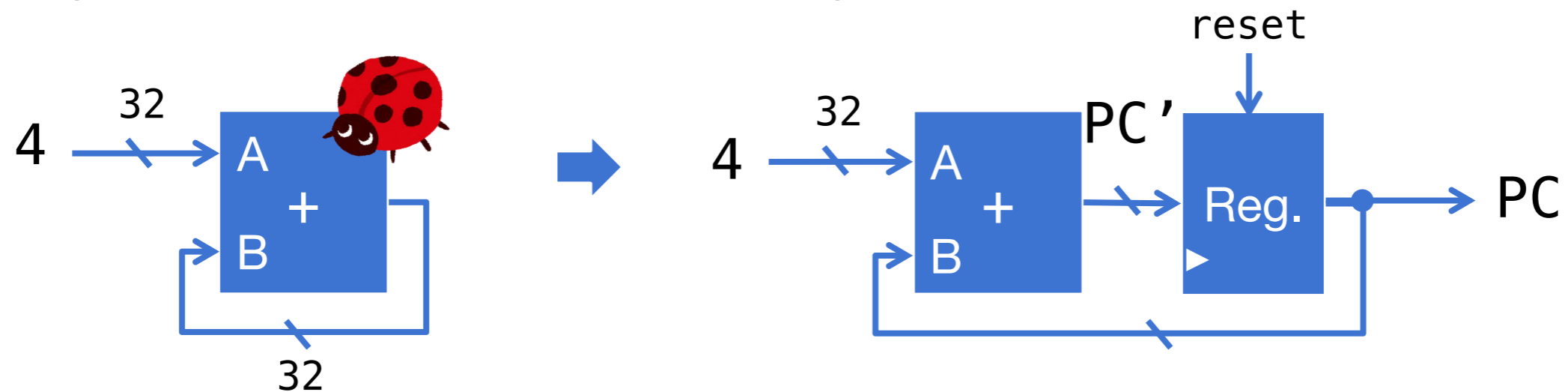
- How do we change values of a specific reg.?



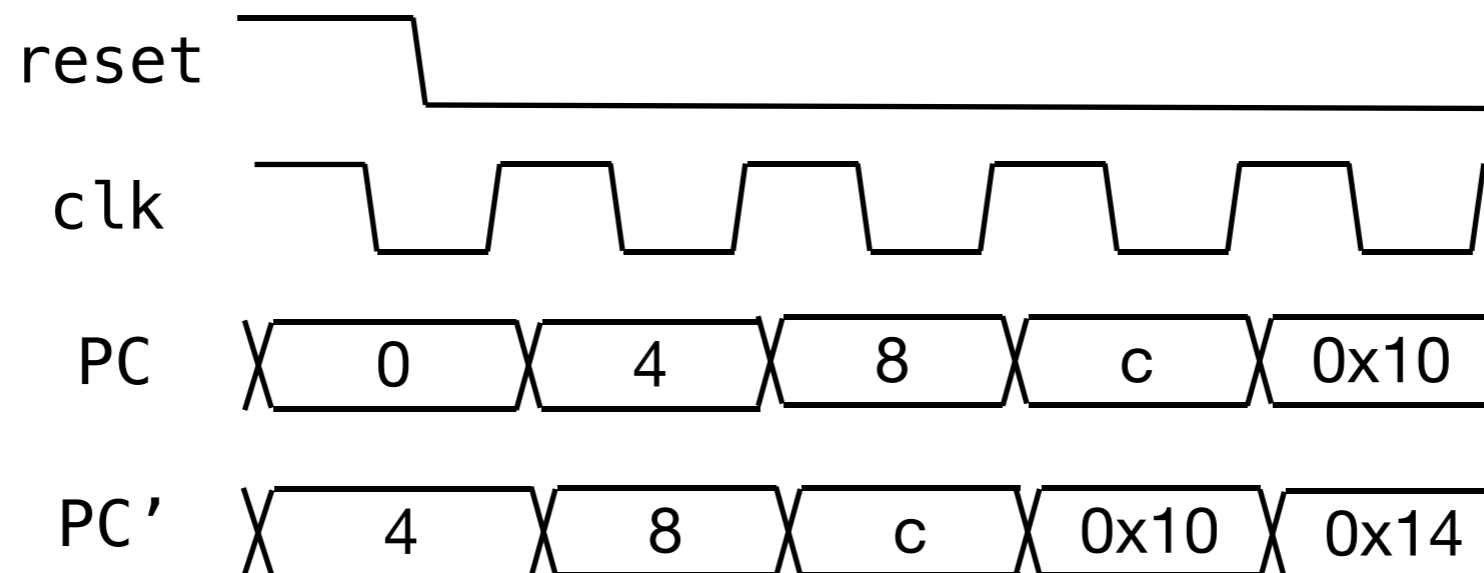
- Reg. file design completed

We have covered PC register previously

- Synchronous digital circuit can have feedback, e.g., iterative accumulator
 - e.g. $PC = PC + 4$ without considering branch or jump

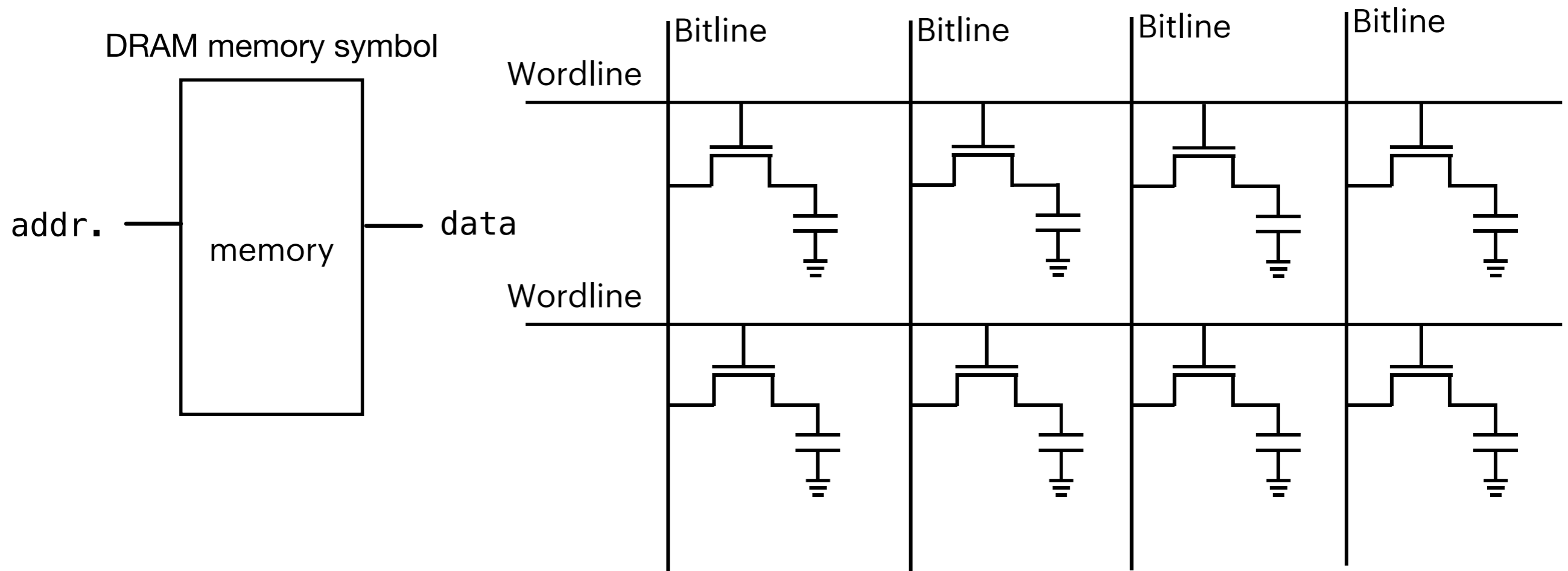


- Timing diagram

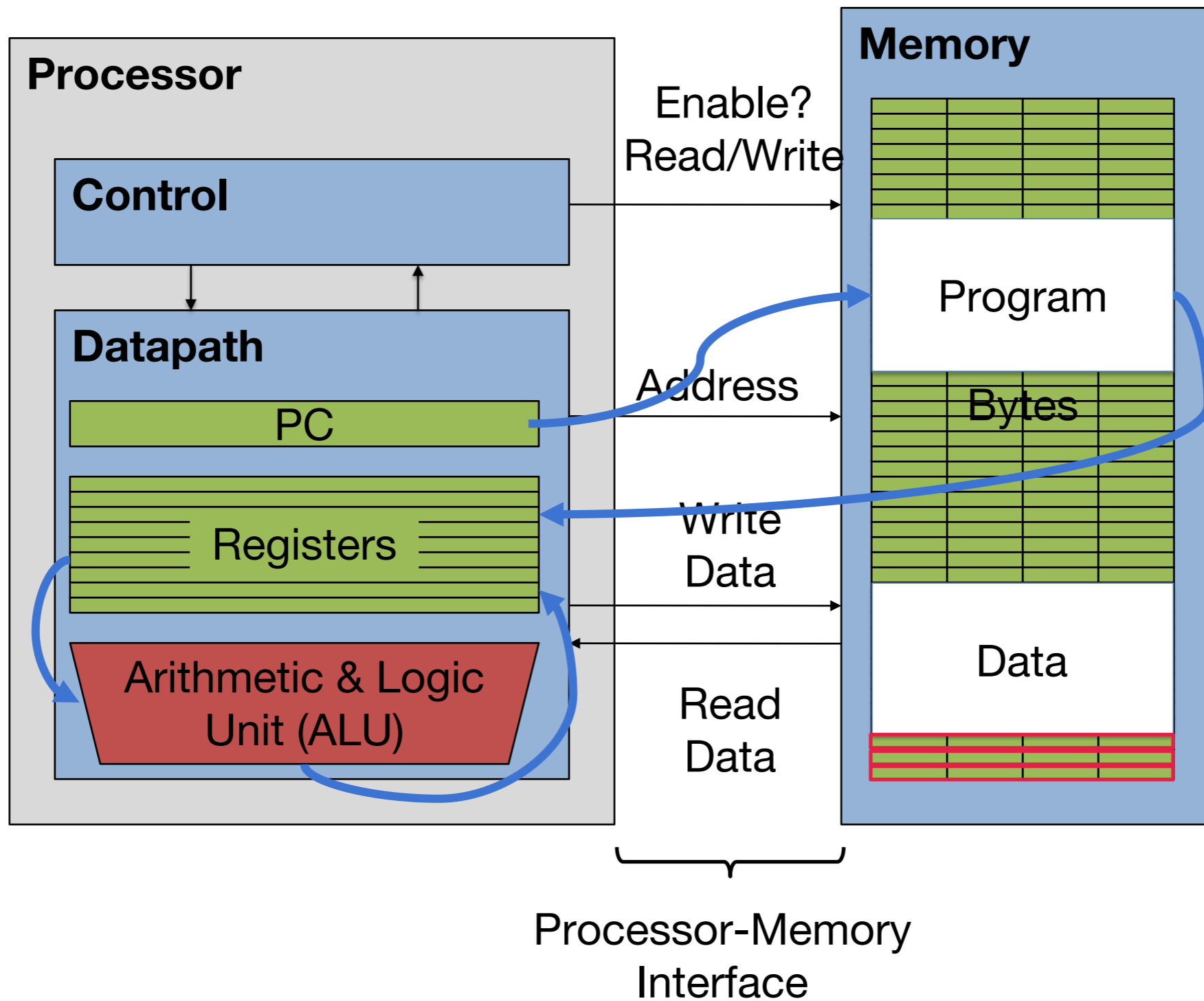


Useful building blocks-Memory

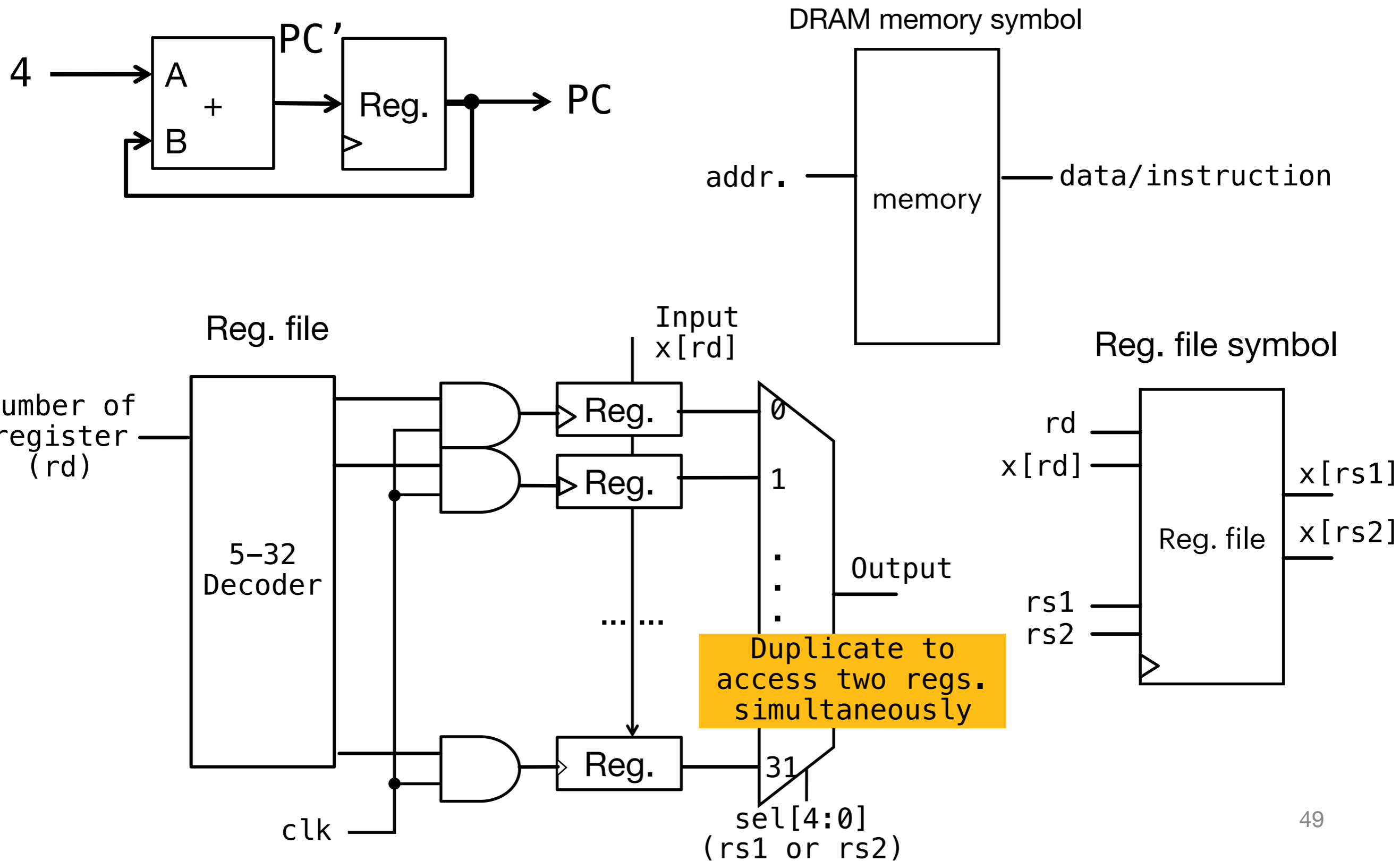
- Memory similar to register file except that the basic cell design is different
- Requires refresh for DRAM



Datapath

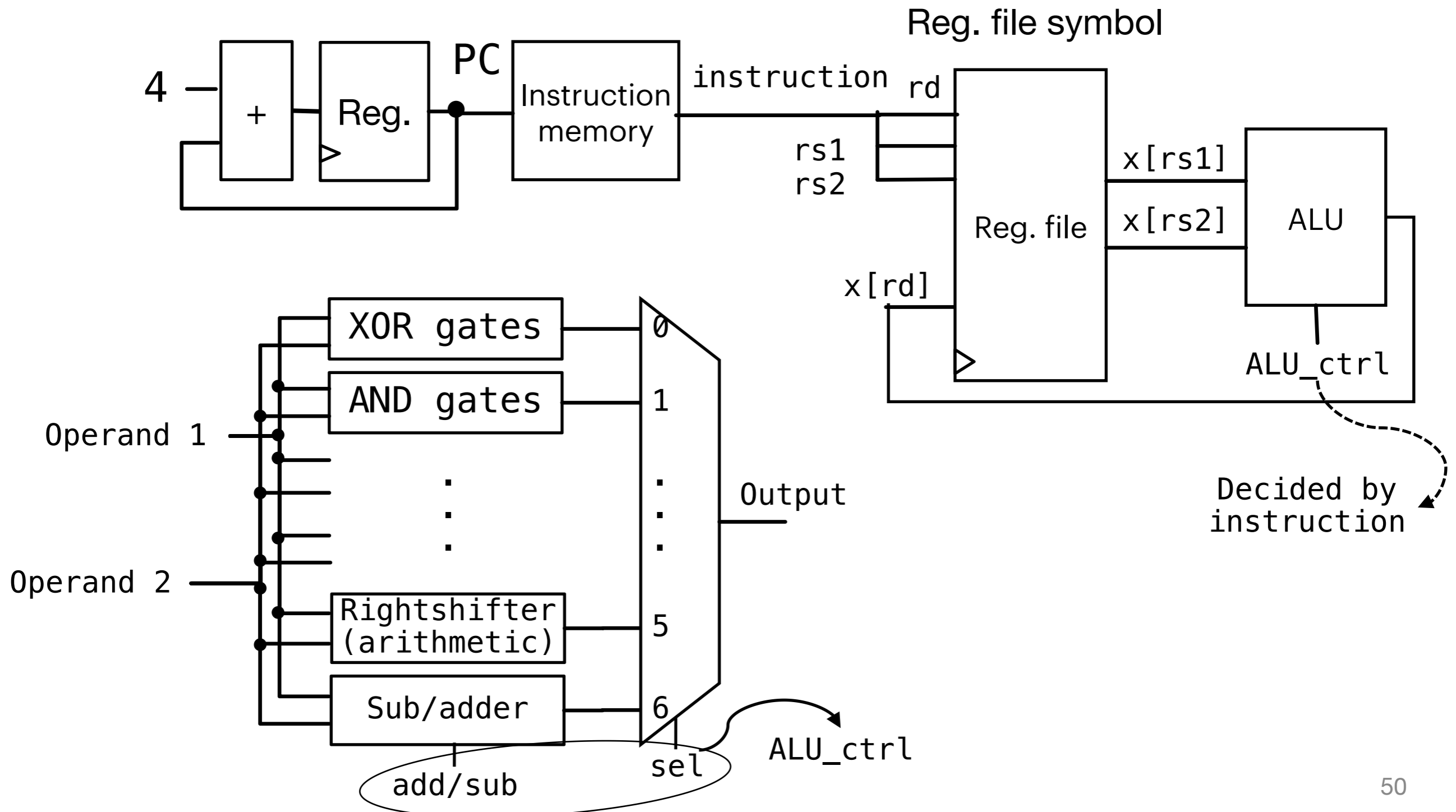


Datapath



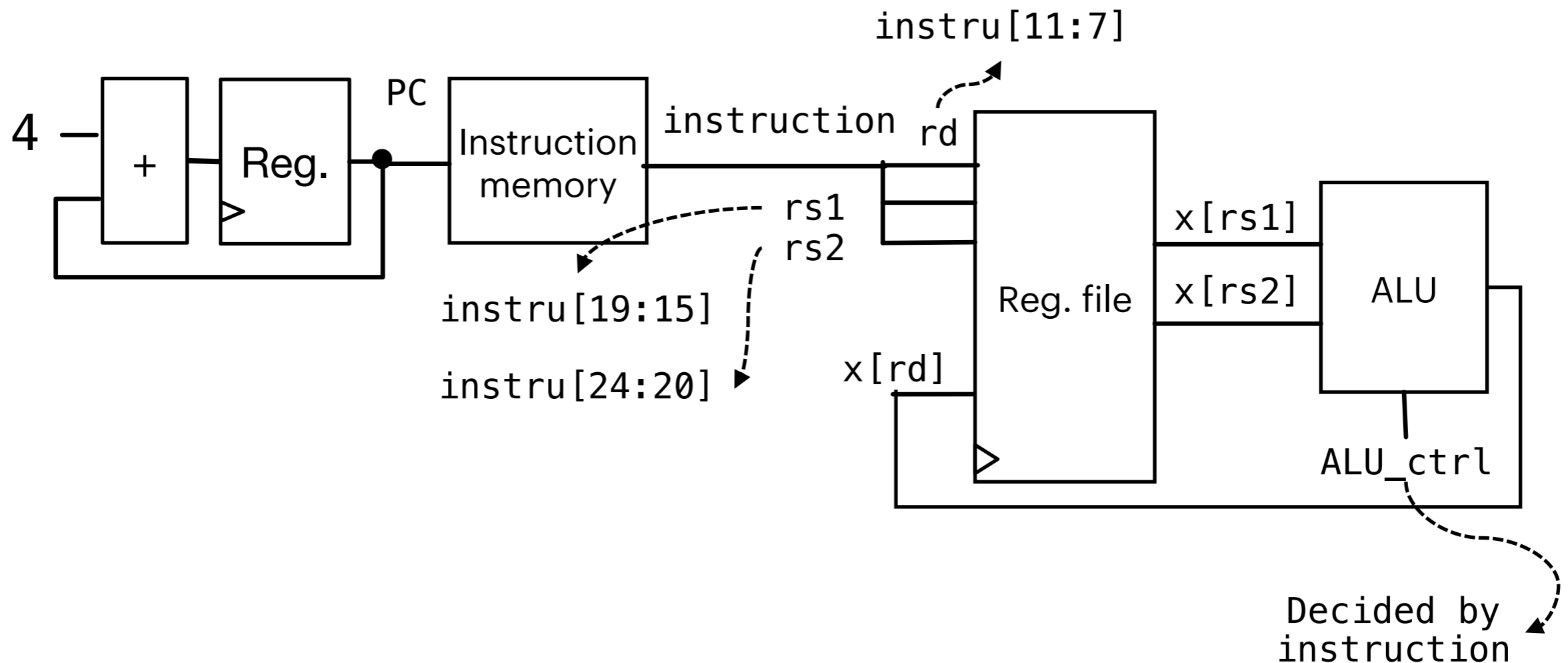
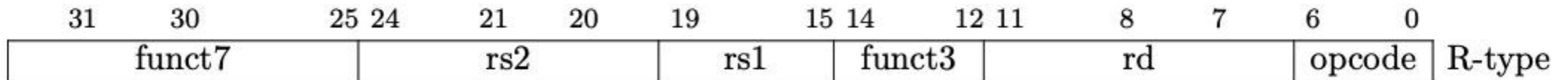
Datapath for R-type

- We have all the building blocks to execute R-type instructions

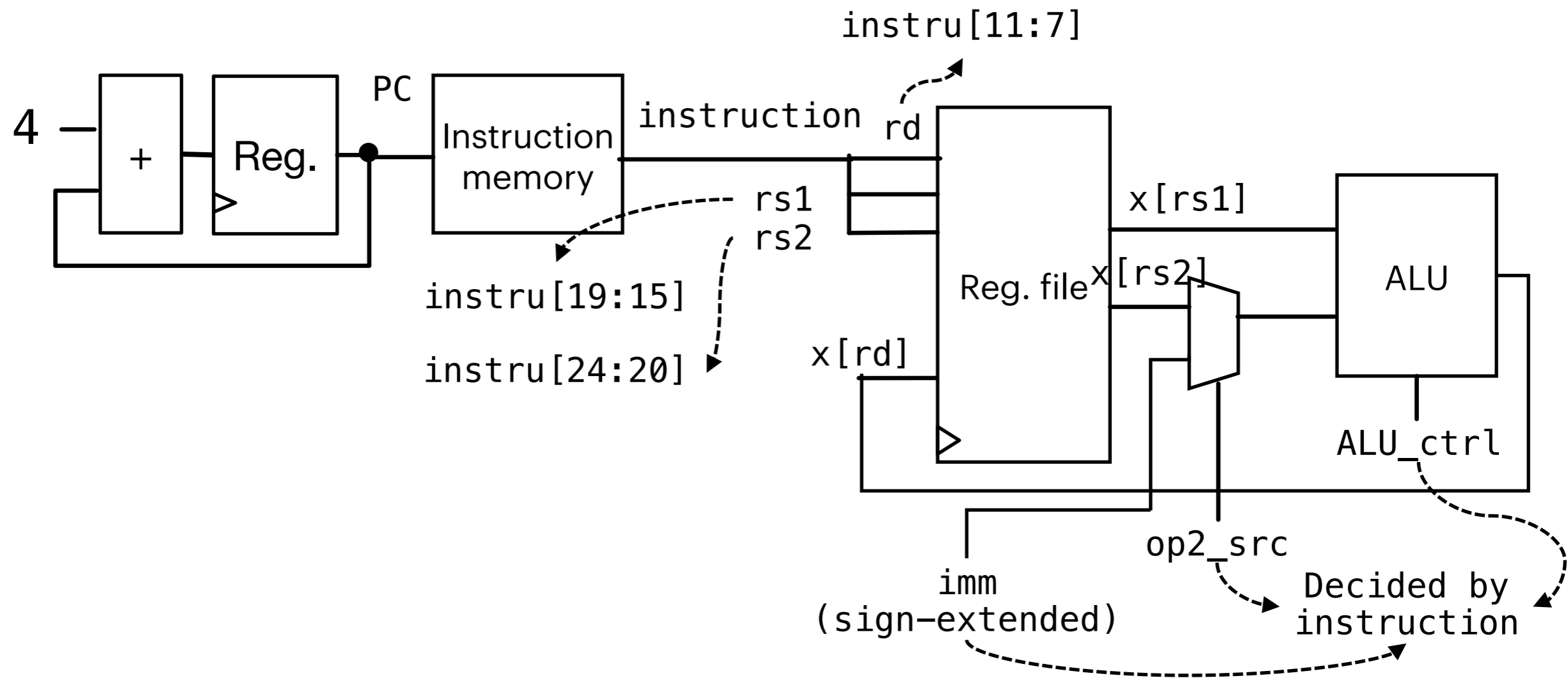
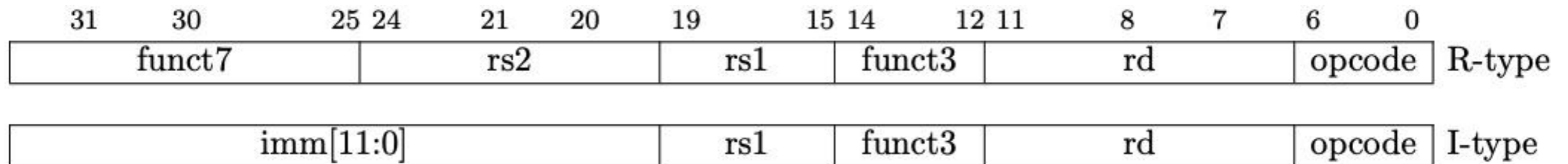


Datapath for R-type

- We have all the building blocks to execute R-type instructions



Datapath for I-type arithmetic and logic



Datapath for more types ...

