Introduction to
Information Science and Engineering

Step1. Computer Architecture

http://archlab.naist.jp/Lectures/ARCH/ca0000/ca0000e.pdf

Download the template and submit through UNIPA.

http://archlab.naist.jp/Lectures/ARCH/ca0000/ca0000e.docx

in http://archlab.naist.jp/Lectures

Goal of today

Q1. What are three main components of a computer?
コンピュータの主要構成要素を3つ挙げよ

Q2. What is the principle that allows even a toy CPU to have five pipeline stages?
おもちゃのCPUでも5段パイプラインステージにできる原理は何か?

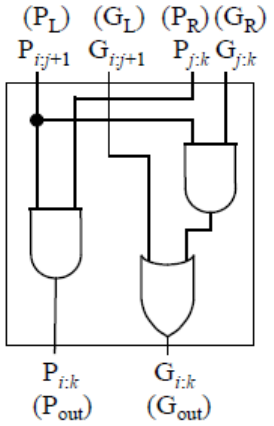
Q3. Can computers calculate 5 billion + 1 correctly?
コンピュータは,20億+1を正しく計算できるか?

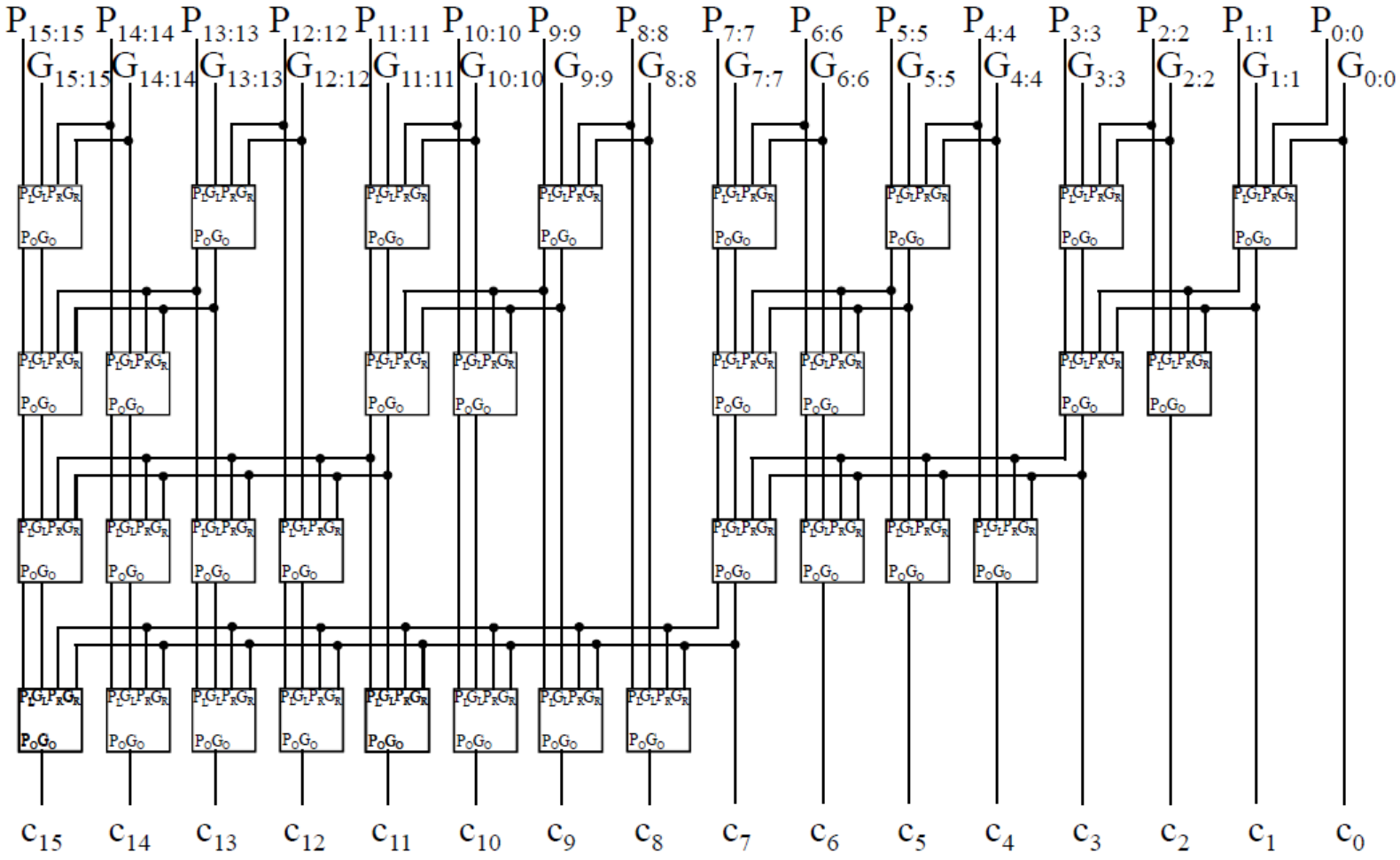
Q4. Your program is correct. But 100 times slower than colleague's program. What should you do?
あなたのプログラムは実行結果が正しい.でも同僚のプログラムより100倍遅い.気付くべき点は?

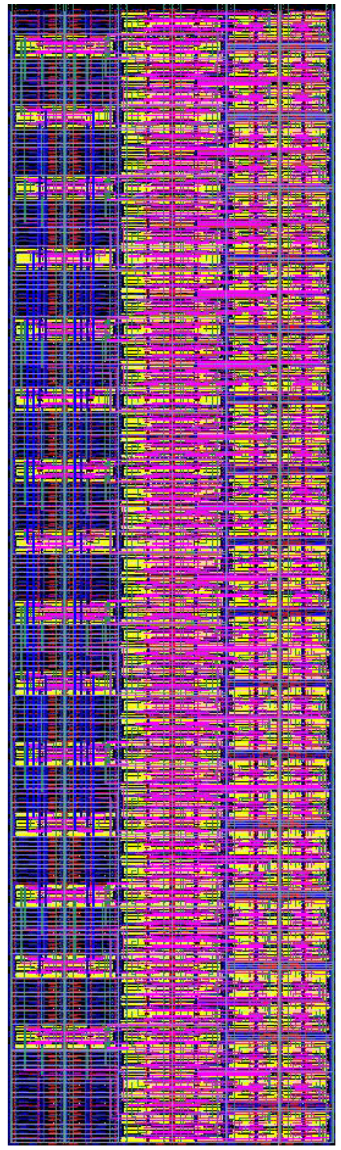
Q5. What can you do for the energy and digital deficit problem derived from depending on imported computers?
海外製計算基盤依存の結果生じたエネルギー・デジタル赤字問題の解決方法は何か?

Arithmetic unit with logical elements

N-bit arithmetic requires Log N stages







Basic units with Logic elements

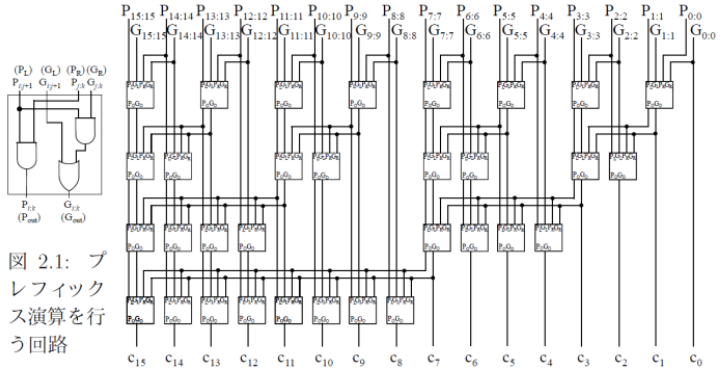
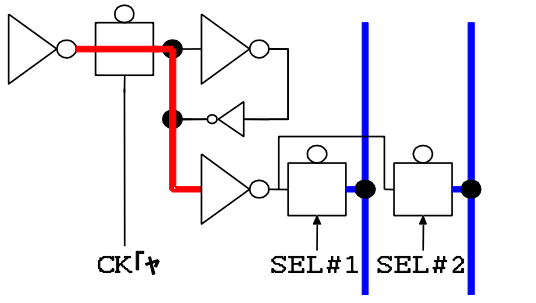
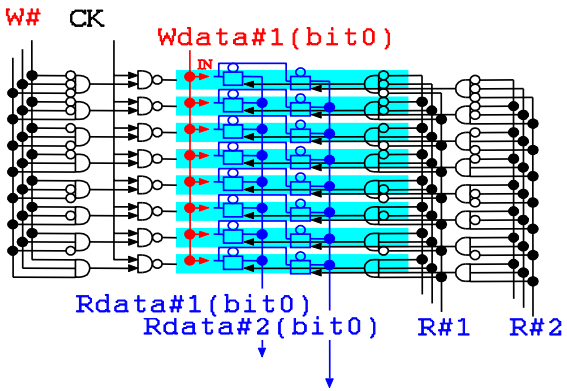


図 2.1: プレフィックス演算を行う回路

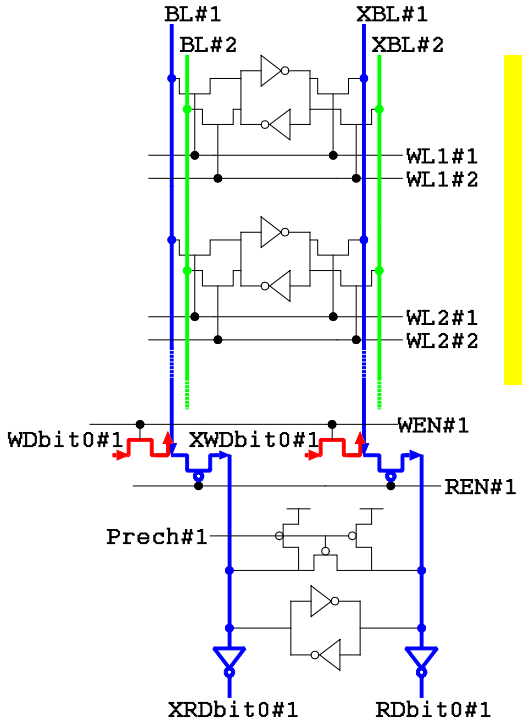
ALU



Flip Flop

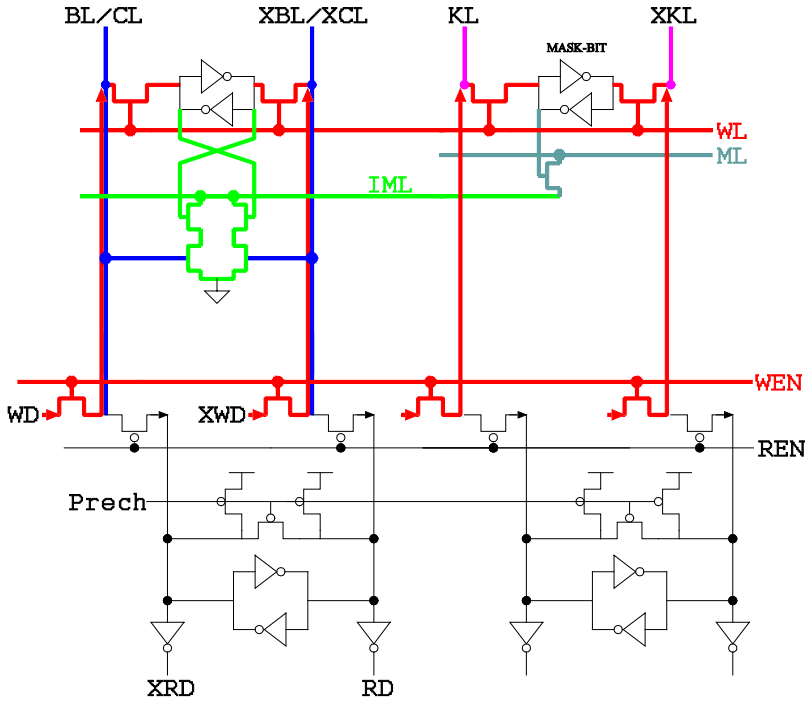


Register File

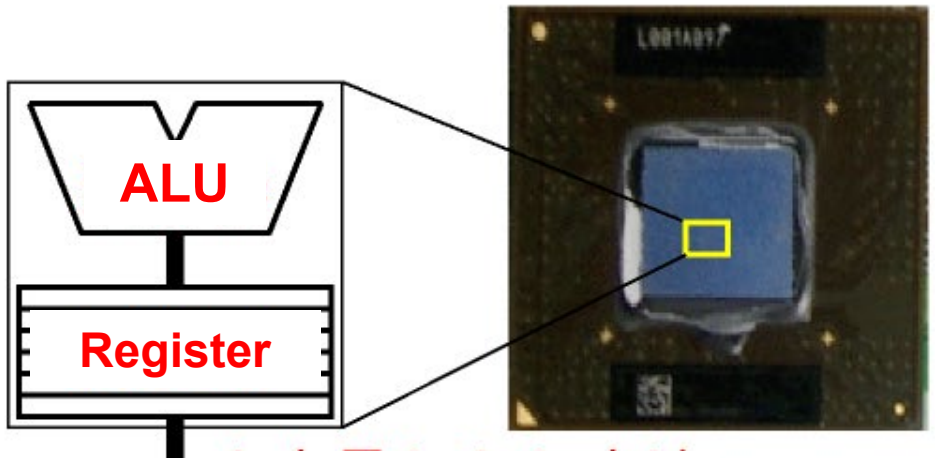


RAM (Random Access Memory)

CAM (Content Addressable Memory)



Computer with Basic units



CPU（Central Processing Unit）

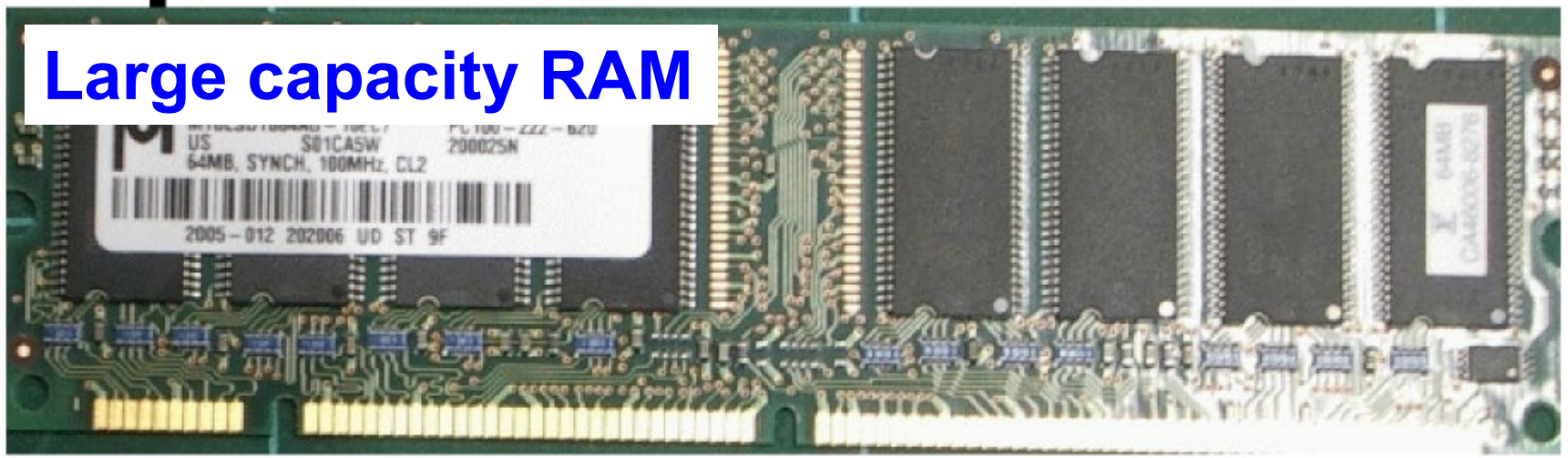
I/O device

Main Memory Unit

Small but Fast

Large but Slow

Large capacity RAM



Goal of today

Q1. What are three main components of a computer?
コンピュータの主要構成要素を3つ挙げよ

Q2. What is the principle that allows even a toy CPU to have five pipeline stages?
おもちゃのCPUでも5段パイプラインステージにできる原理は何か?

Q3. Can computers calculate 5 billion + 1 correctly?
コンピュータは,20億+1を正しく計算できるか?

Q4. Your program is correct. But 100 times slower than colleague's program. What should you do?
あなたのプログラムは実行結果が正しい.でも同僚のプログラムより100倍遅い.気付くべき点は?

Q5. What can you do for the energy and digital deficit problem derived from depending on imported computers?
海外製計算基盤依存の結果生じたエネルギー・デジタル赤字問題の解決方法は何か?

Computer = calculator with program

1. **Sample C-source**

```
% vi sample.c
main()
{
  printf("Hello.\n");
}
```

2. **C Compiler generates executables**

- **Preprocessor, Compiler, Assembler and Linker are called**
- sample.c ⇒ cxxx.i ⇒ cxxx.s ⇒ cxxx.o ⇒ sample

```
% gcc -v sample.c -o sample
…/cpp …    sample.c      …/cxxx.i
…/cc1 …    …/cxxx.i -o   …/cxxx.s
…/as  …    …/cxxx.s -o   …/cxxx.o
…/ld  …    …/cxxx.o -o   sample
```

3. **Execution**

```
% ./sample
Hello.
```

Source code to instructions

Source code

```
int R1[100];                    /* reg r1: top address of array R1[] */
int R2;                         /* R2: index for array R1[] */
R1[1] = R1[1]+8;                /* add 8 to second element of R1[] */
R1[R2]= R1[1]-R1[R2];           /* reg r2:  R2 */
```

Instructions

```
ld  r1,4,r4          r4 ← mem[r1+4]
add r4,8,r8          r8 ← r4 + 8
st  r1,4,r8          Mem[r1+4] ← r8
ld  r1,r2<<2,r5      r5 ← mem[r1+r2*4]
sub r8,r5,r9         r9 ← r8 – r5
st  r1,r2<<2,r9      Mem[r1+r2*4] ← r9
```

Computer repeats Fetch, Decode, Execution, Memory-access and Write-back.

Typical functional decomposition

- 1. Fetch

  Fetch an instruction from PC address of main memory
- 2. Decode

  Decode the instruction and read data from register

  Prepare control signals for following stages
- 3. Execute

  Execute arithmetic, logical, shift or other operation

  For load/store instruction, calculate effective address (add or sub)
- 4. Memory

  For load instruction, read data from main memory to register

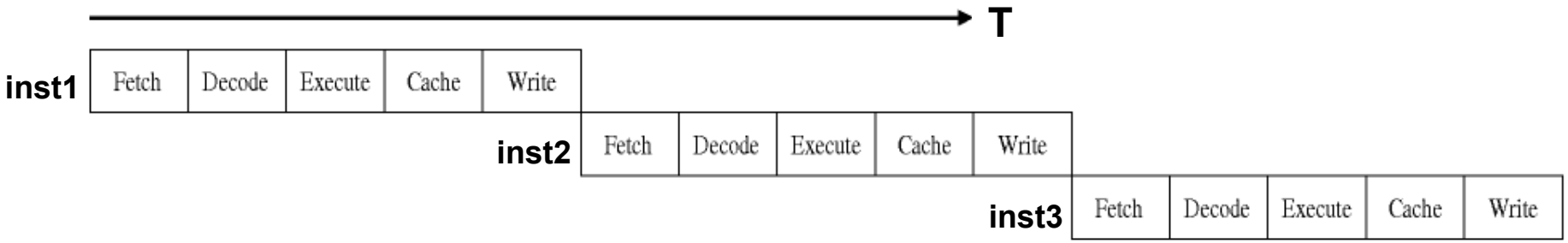
  For store instruction, write data from register to main memory

  Other instruction, do nothing
- 5. Write

  Store result of execution or load data into a register

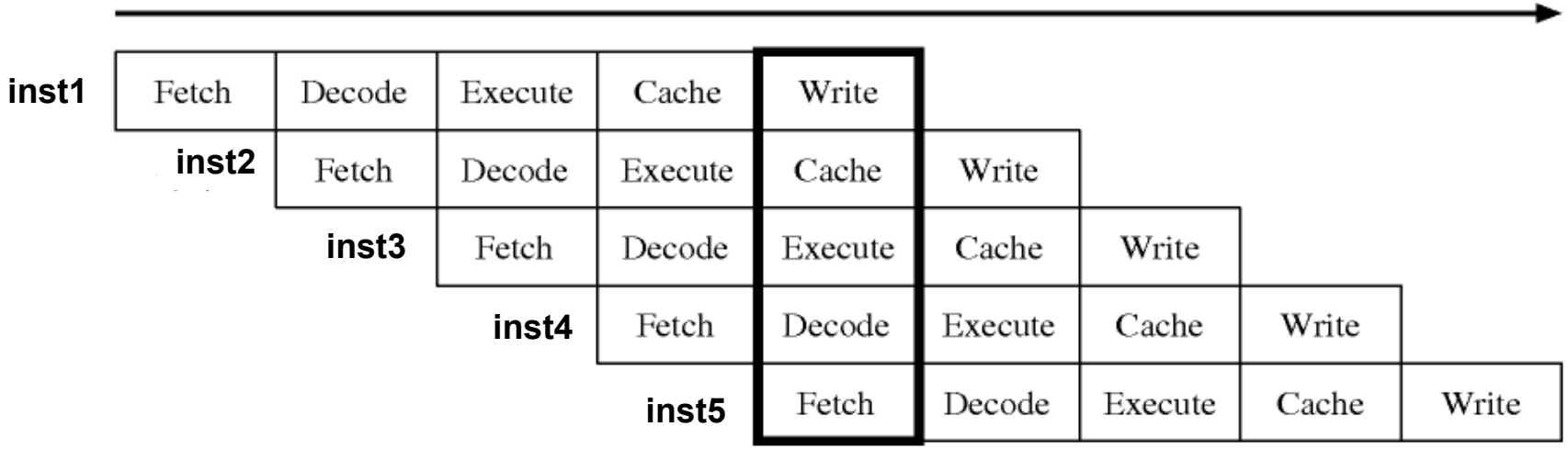
Pipeline execution from fetch to write

Just do it sequentially … like ancient 8bit CPU (no cache)



Fast execution by stage parallel execution

- Since temporal storage (pipeline registers) is required, the improvement is smaller than the number of stages
- Even slower, if execution time of each stage is not balanced



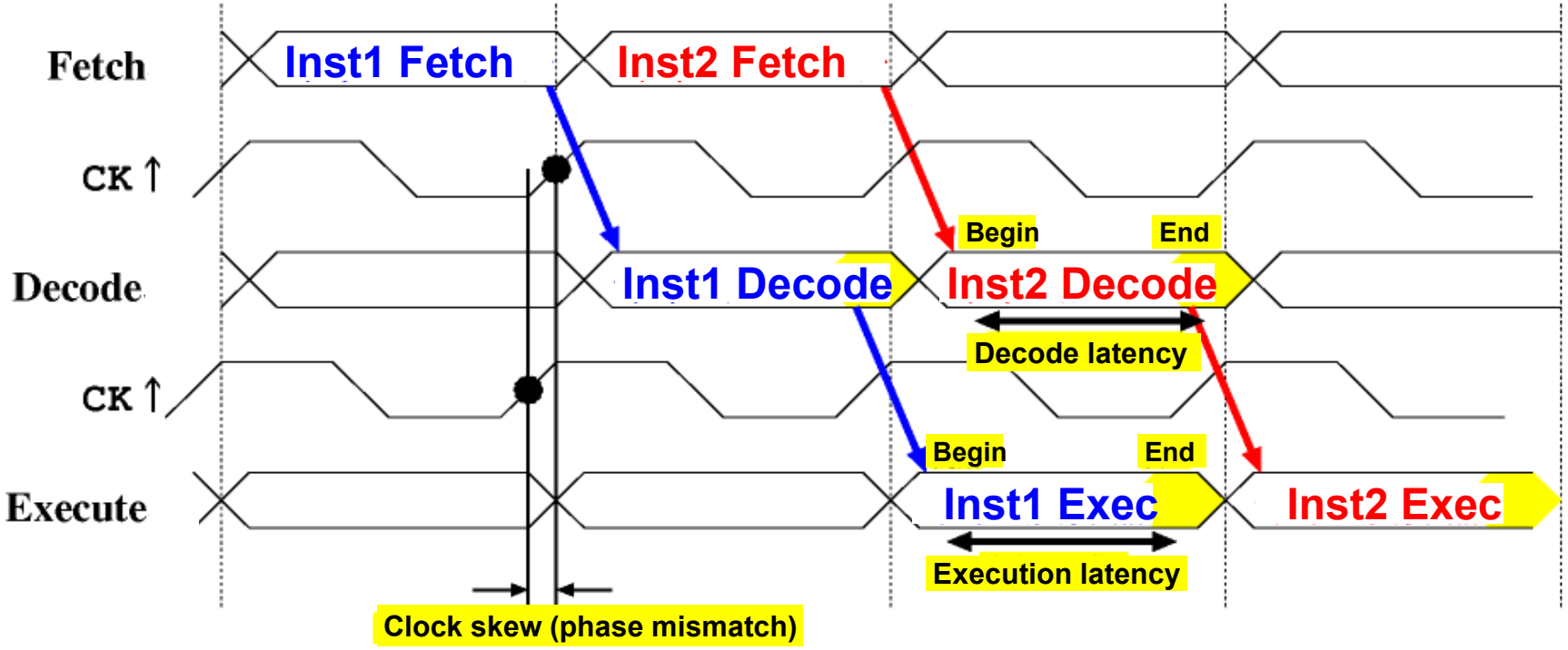
Operation of master slave latch

When CK↑, each pipeline stage starts operation

- **If result of previous stage is arrived, propagation is blocked by latch**
- **If result is not arrived when CK↑, propagation is failed**

  Therefore, to supply higher frequency clock without latency improvement causes malfunction

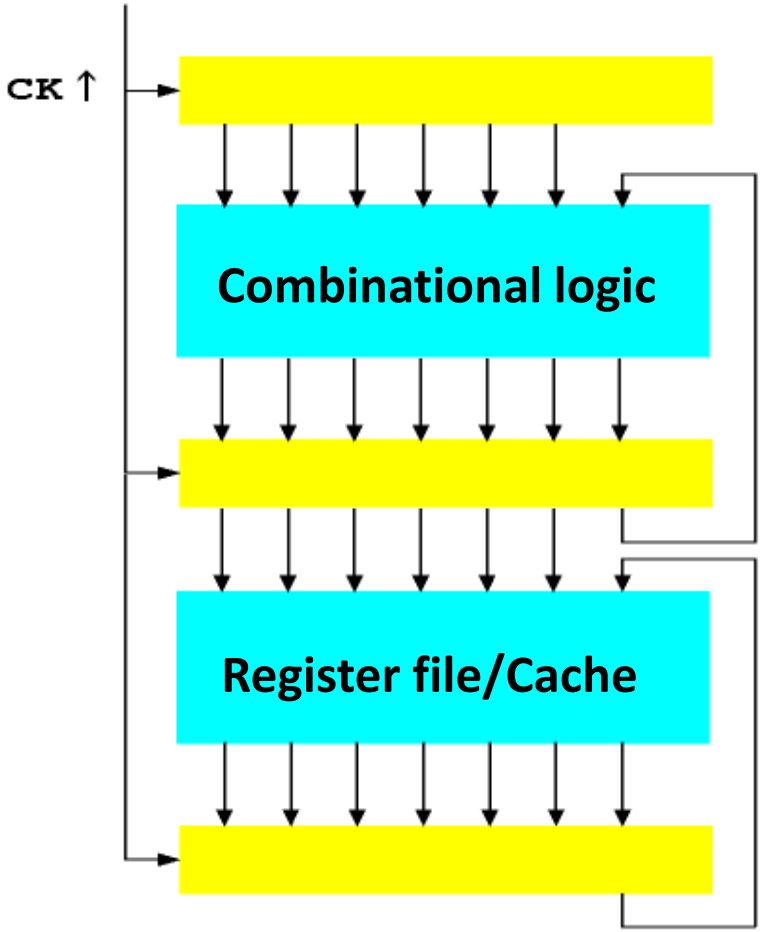
  Clock skew is also common reason of malfunction



Locate pipeline registers between stages

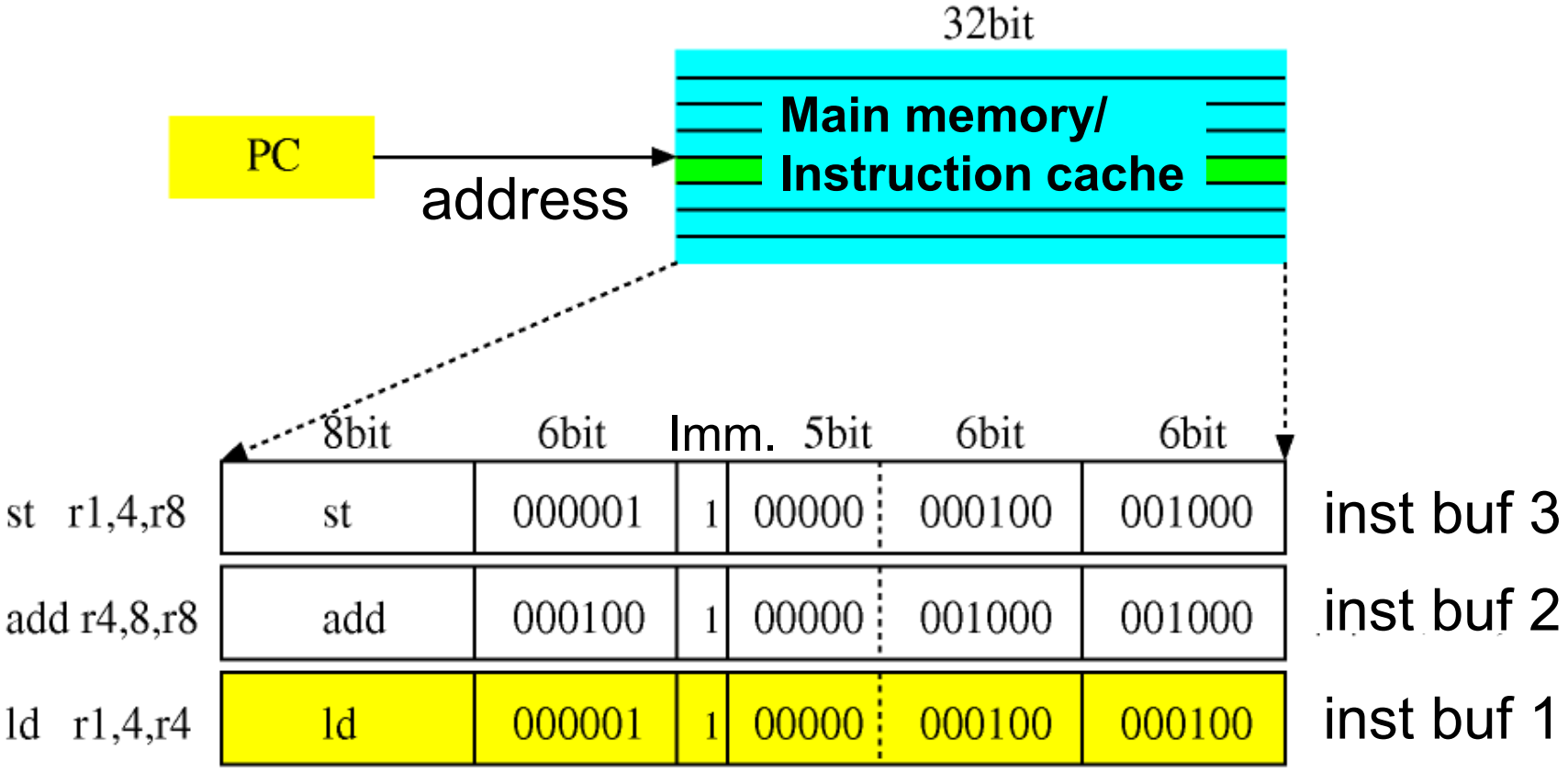
Clock and pipeline register

- By connecting output to input of previous stage, information can be hold
  Ex. Use +4 logic as combinational logic to make program counter



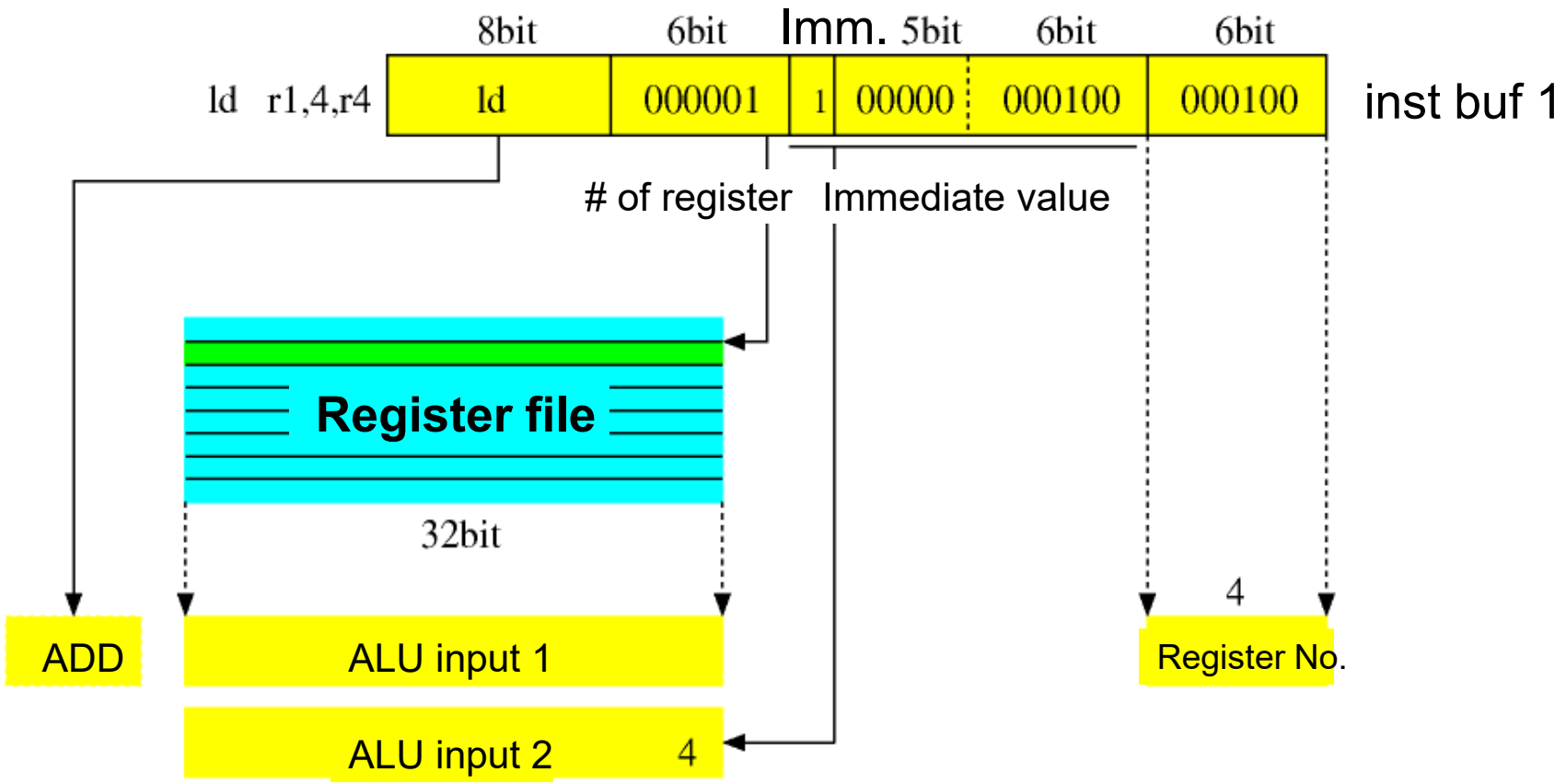
Fetch – Fetch an instruction

Read PC address of Main memory and write result to instruction buffer



Decode – Decode instruction

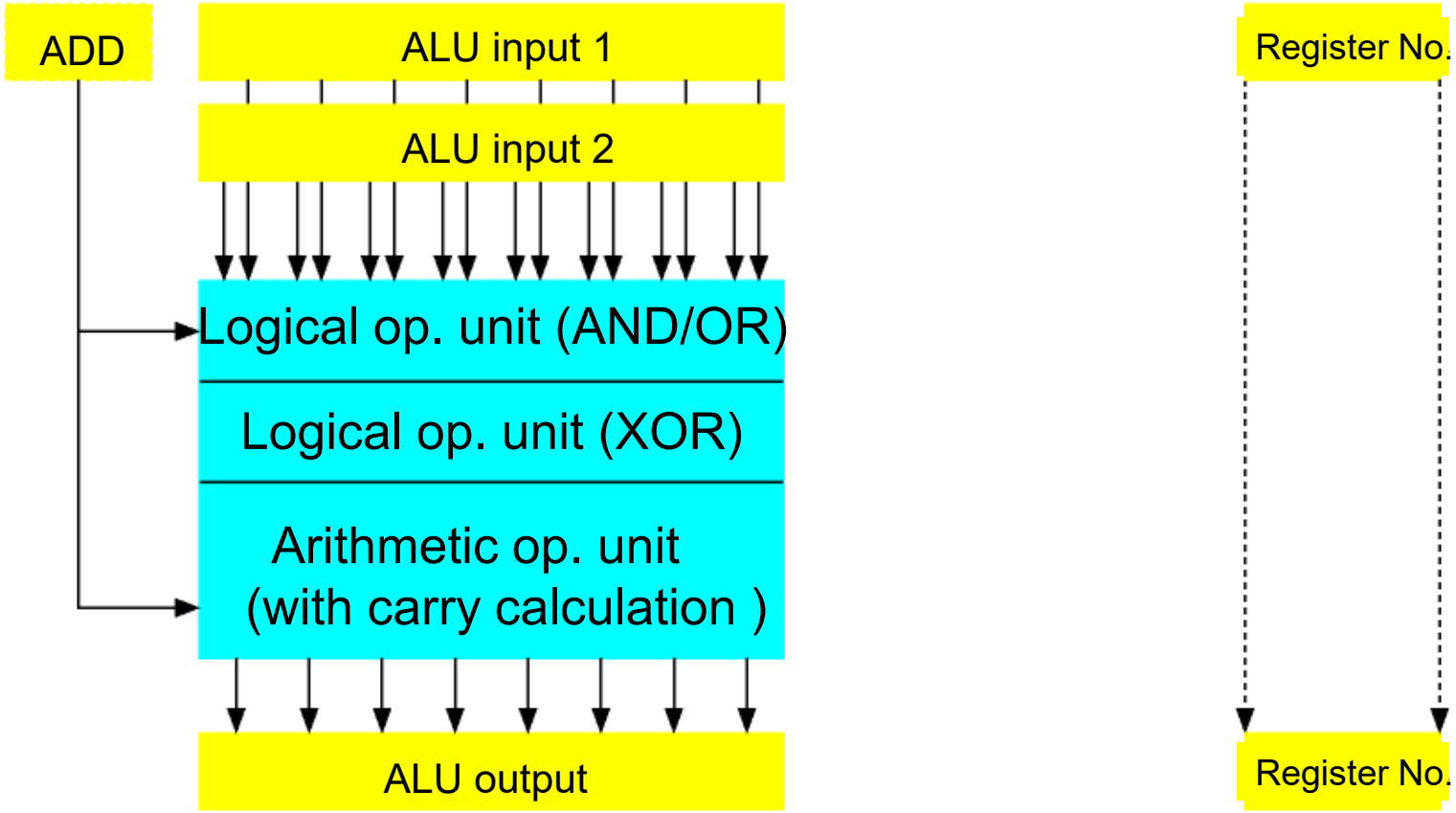
Decode instruction then prepare control signals for ALU, input data, output register number



Execute

Do operation that is specified by control signal, result is stored temporally

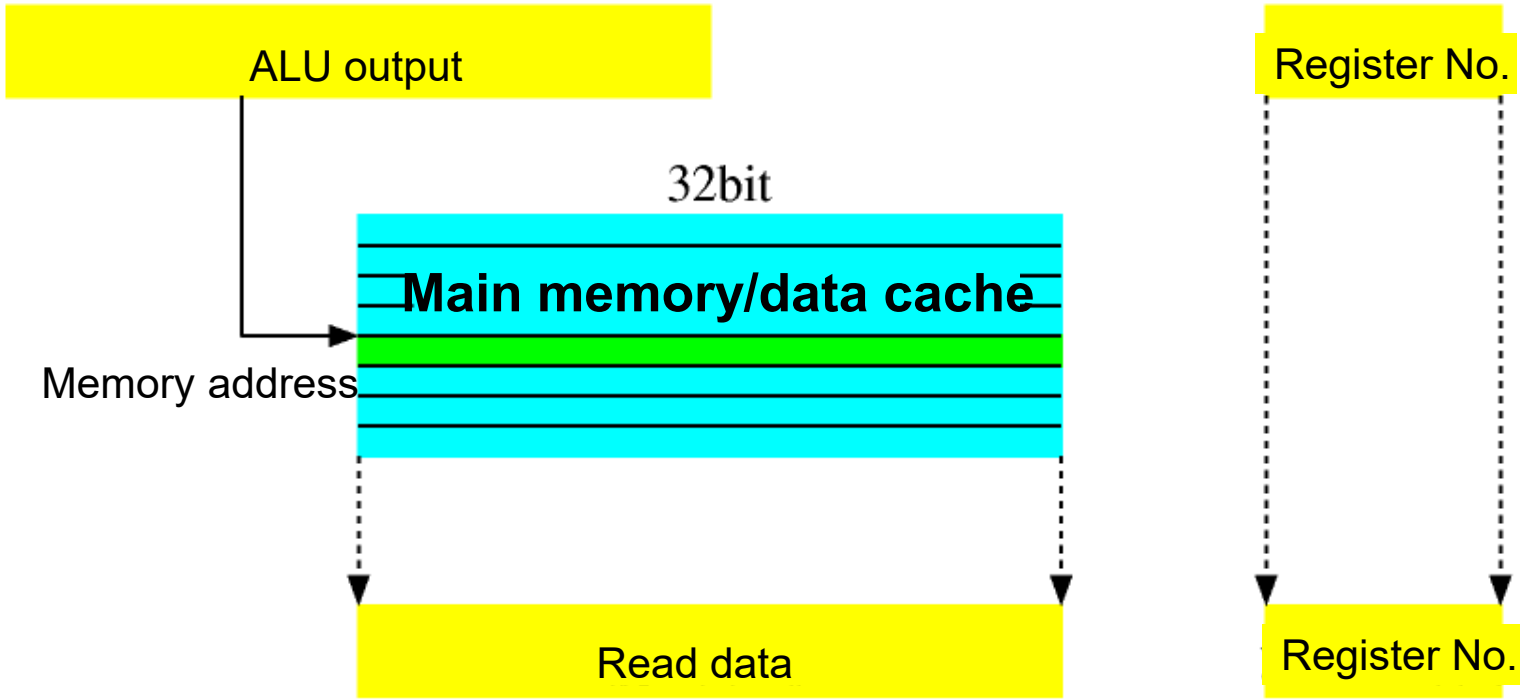
- **Do address calculation for load/store instructions**



Cache – Read operand

Access memory with calculated address and store result temporally

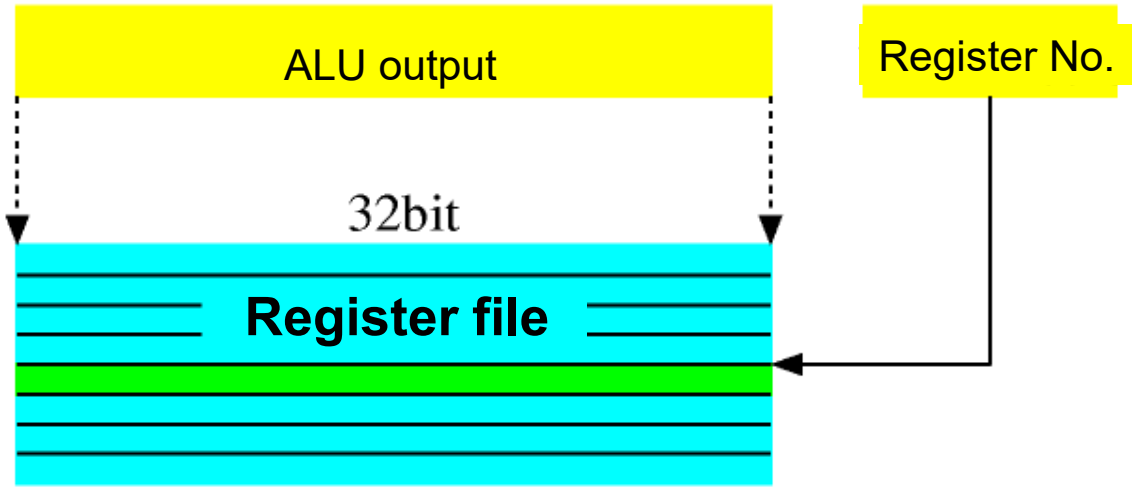
- **If virtual address cache, TLB is in this stage**
- **If store buffer is implemented, it is also in this stage**



Write – Write result

Write result to register file

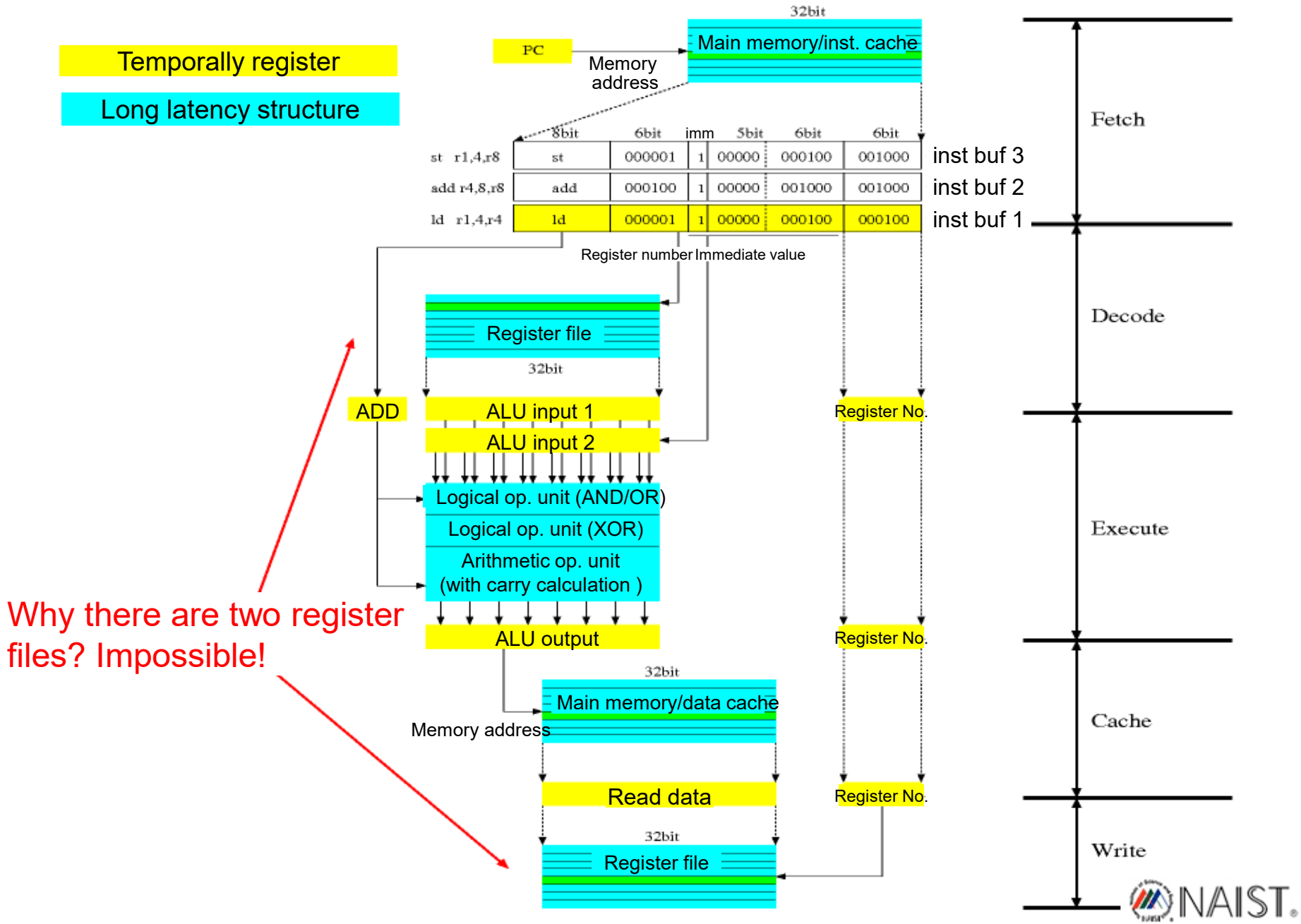
- **Note that there is register file in Decode stage**
- **If it is composed of latch, half cycle READ/WRITE is possible**
- **If it is composed of memory, half cycle operation is not possible**



Simply connect them together

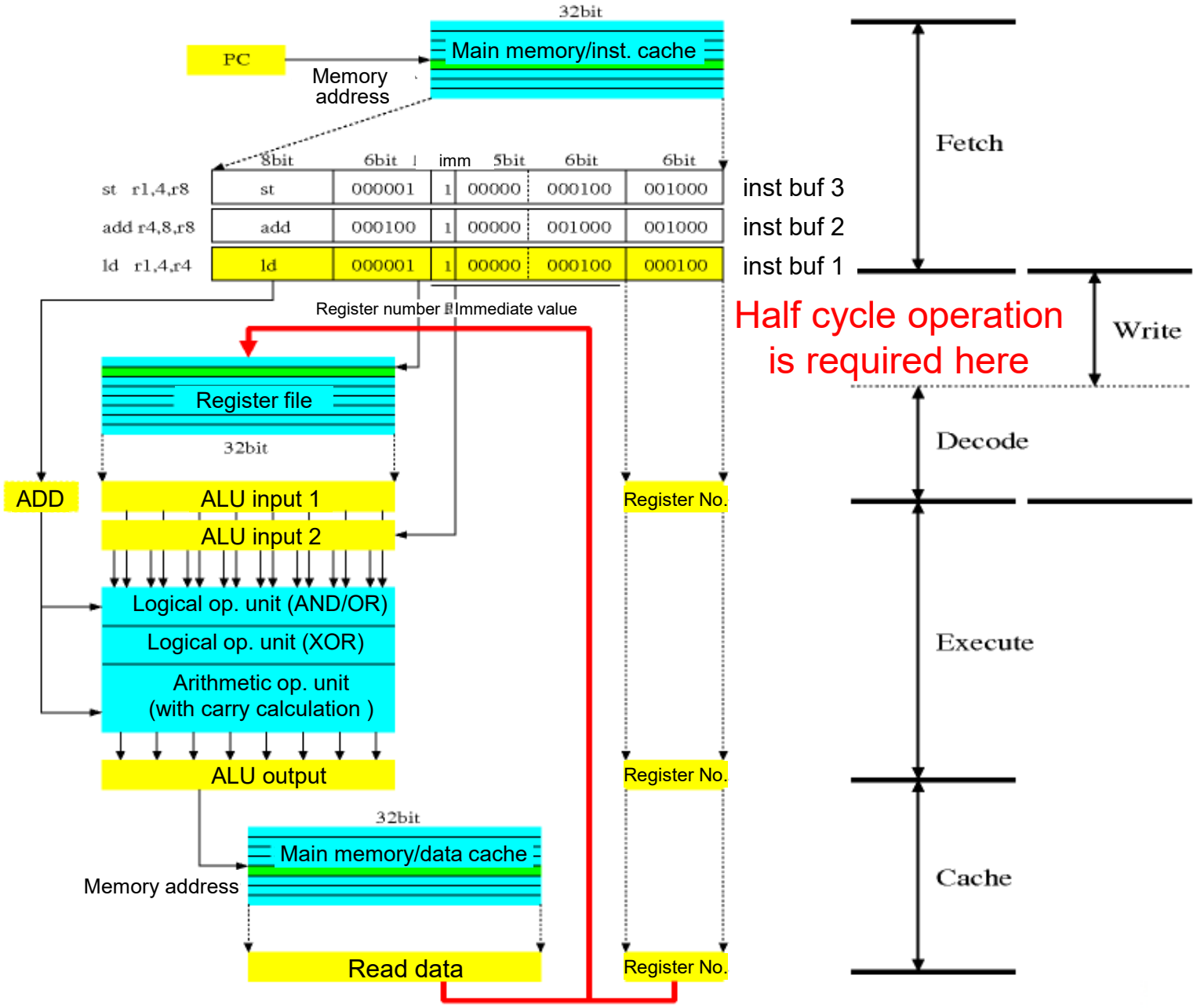
Temporally register

Long latency structure



Why there are two register files? Impossible!

Connect them correctly (Decode and Write are in one cycle)



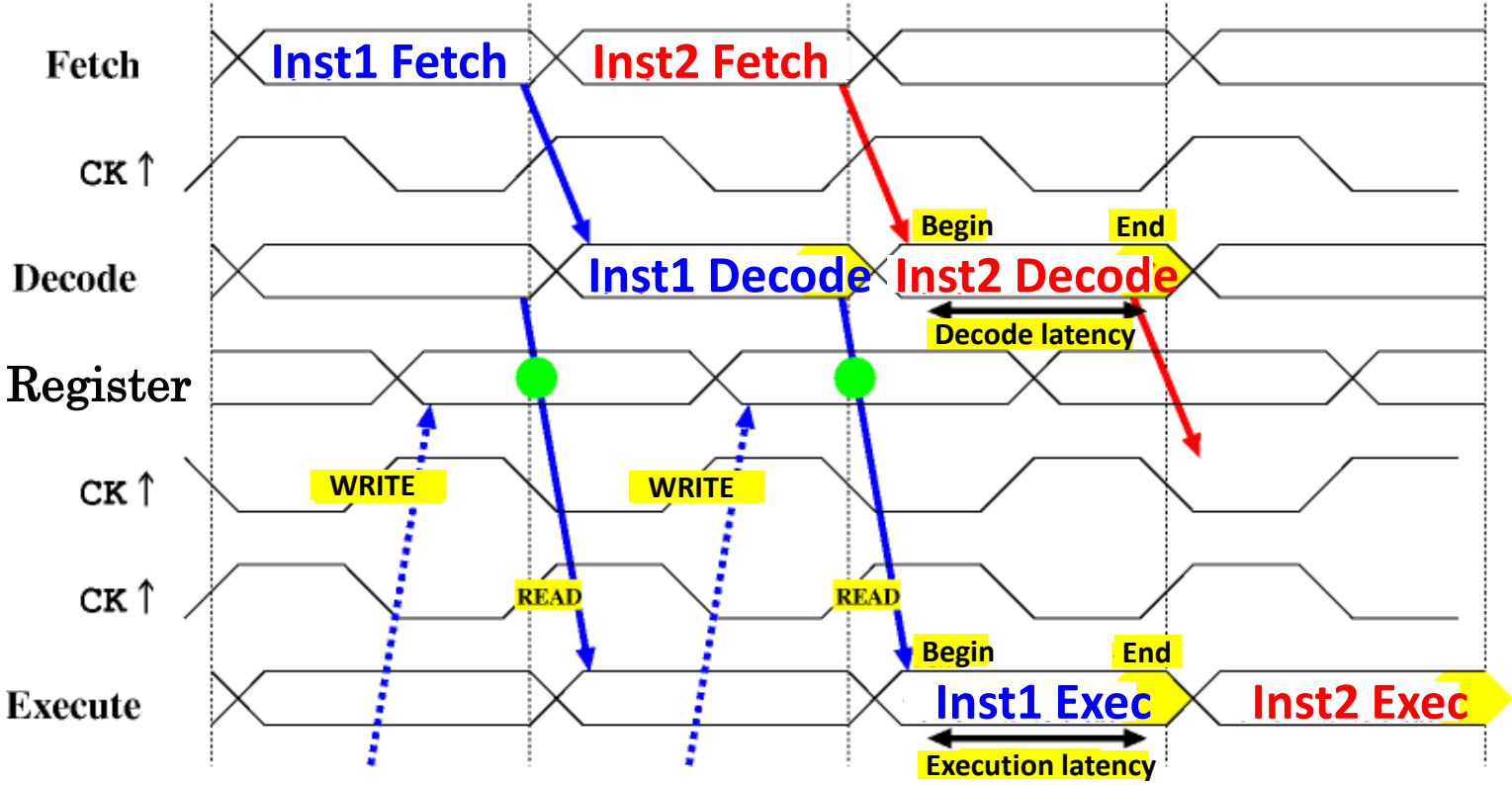
Antiphase operation of pipeline register and register file

If register file is composed of latch, half cycle READ/WRITE are possible

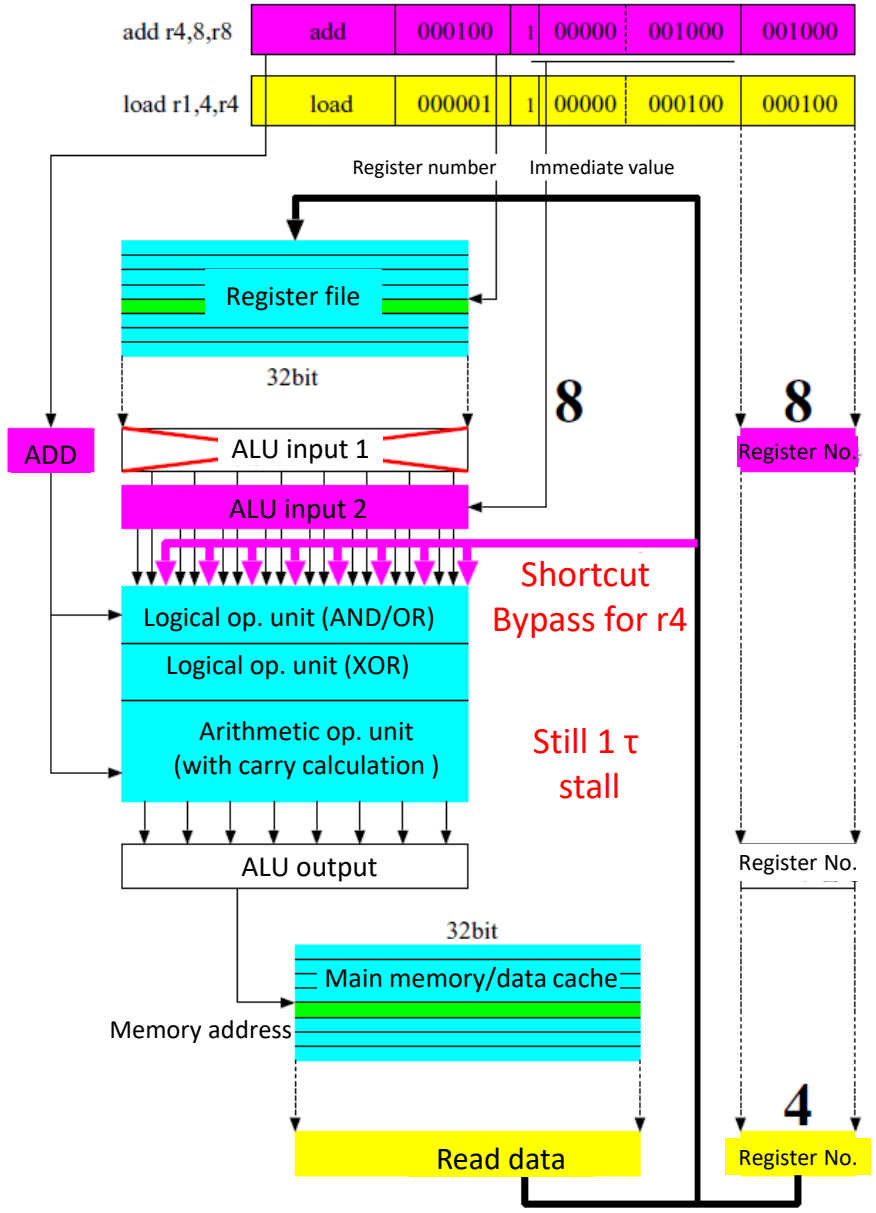
- **Write in first half cycle and read in second half cycle**
- **Register bypassing (explain later) can be omitted**

If register file is composed of memory, half cycle operations are not possible

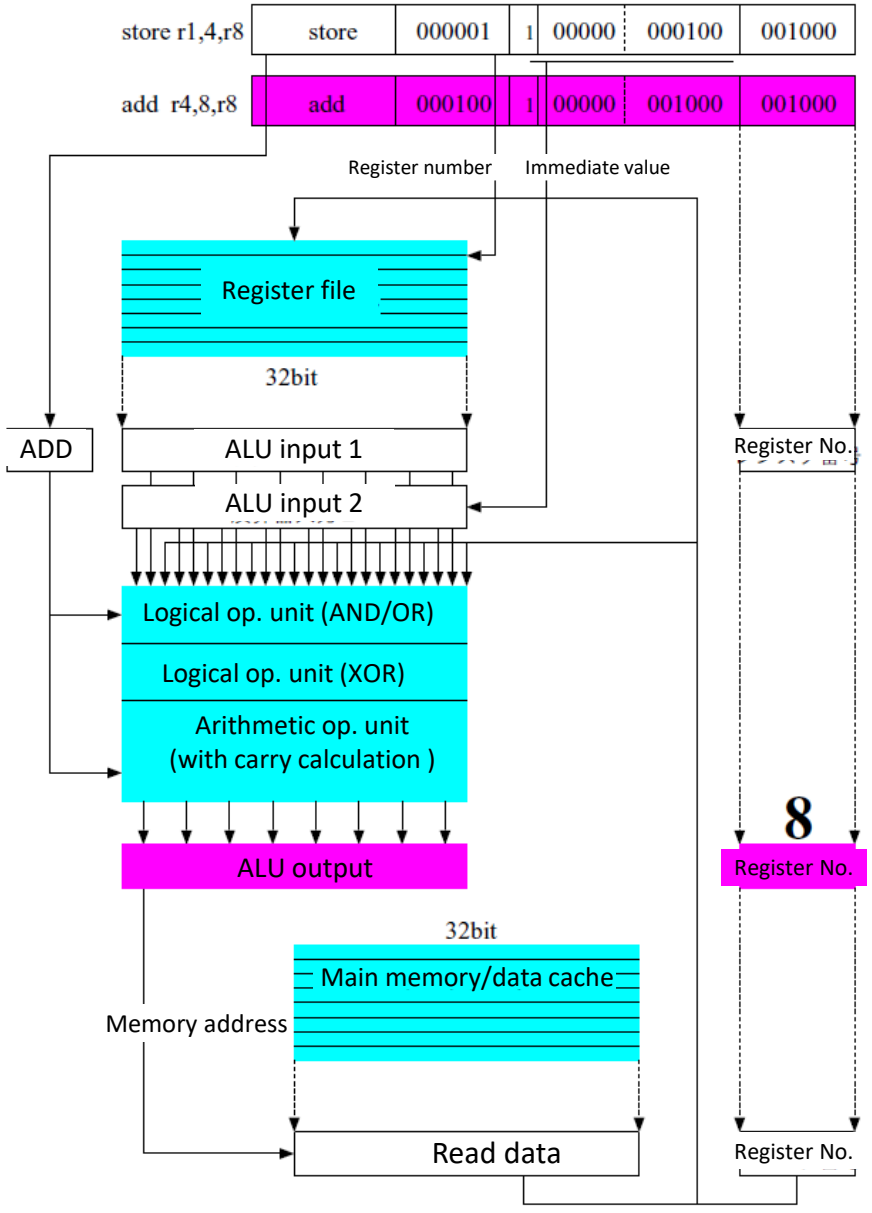
- **Register bypassing (explain later) is required**



Load then add (ld-use penalty if dependency is exists)

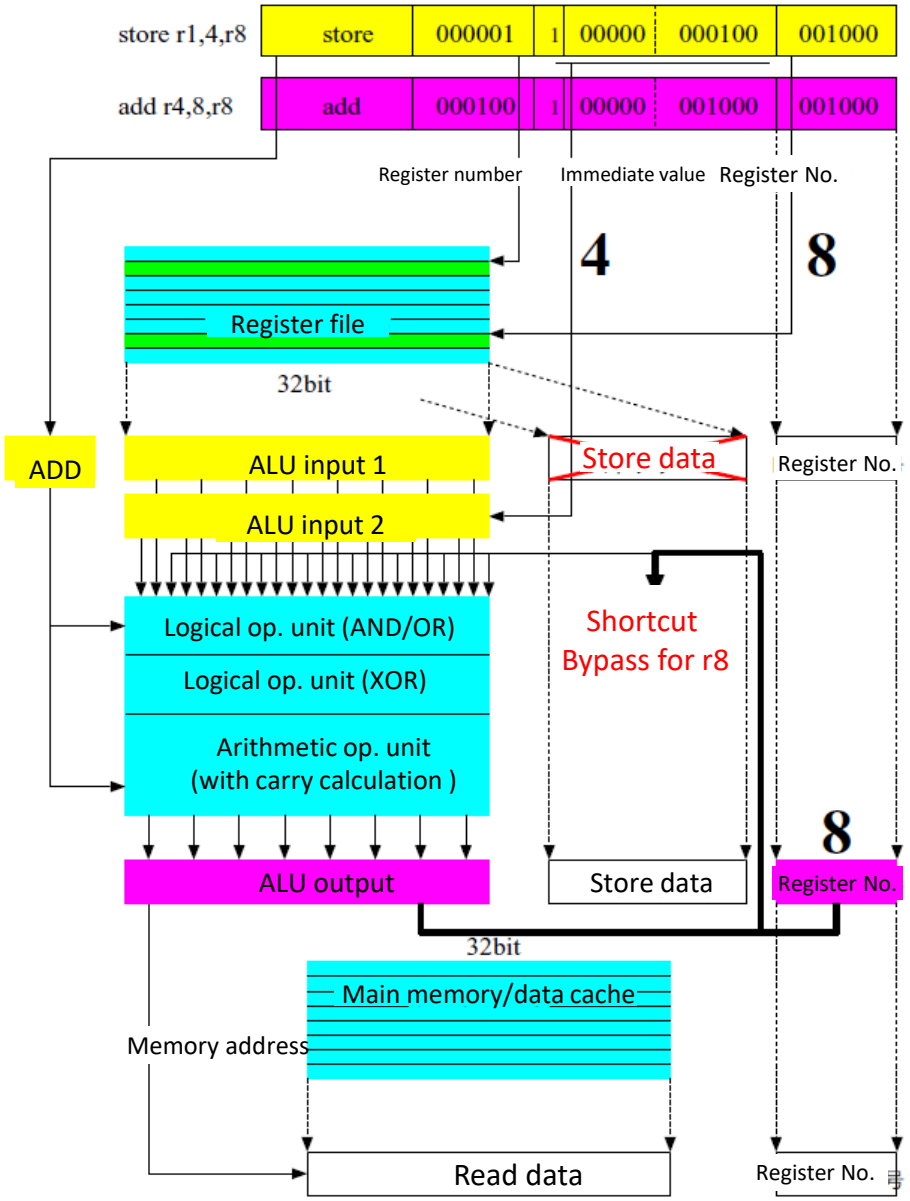


When load is done

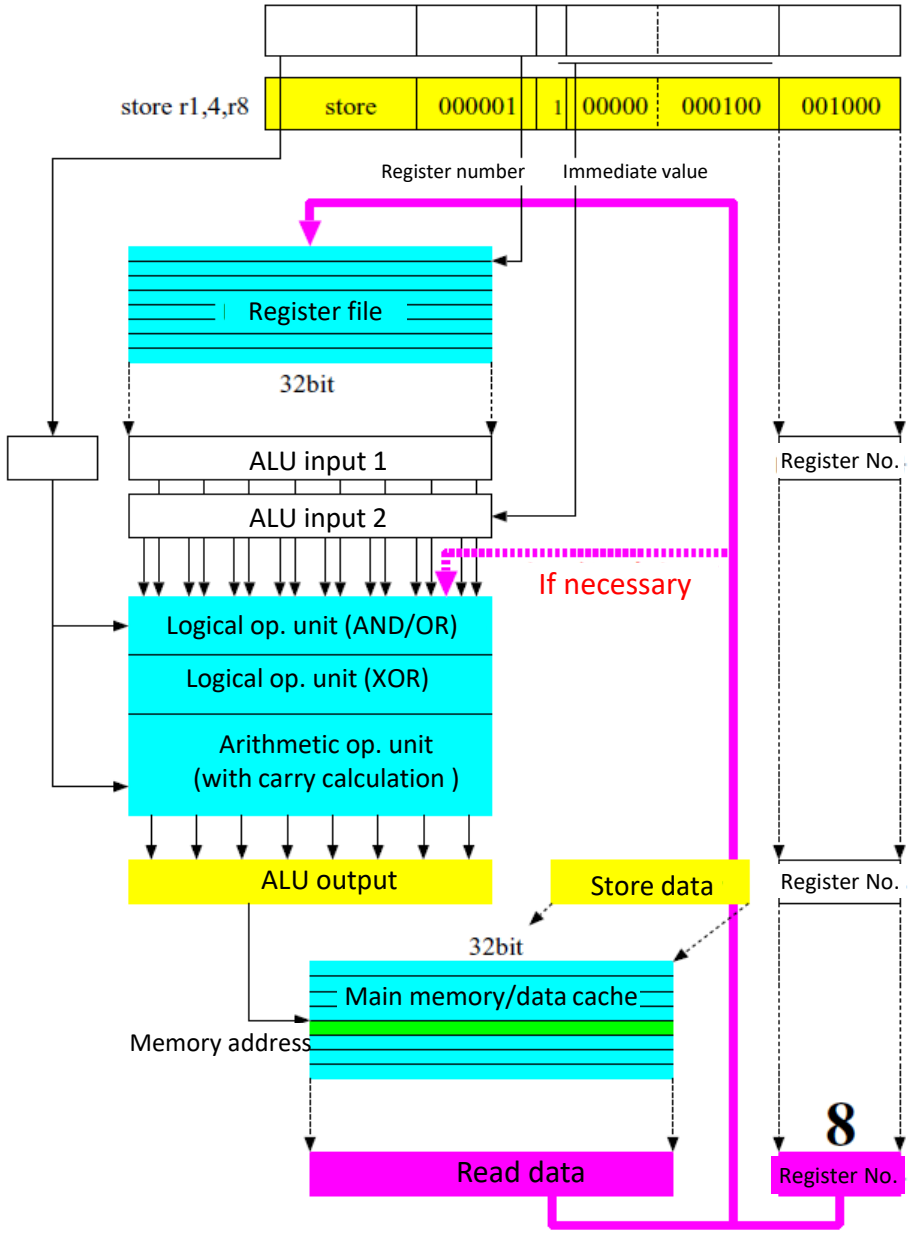


When execution of add is done

Add then Store (no penalty by bypassing)



When execution of add is done



When add is completed

Goal of today

Q1. What are three main components of a computer?
コンピュータの主要構成要素を3つ挙げよ

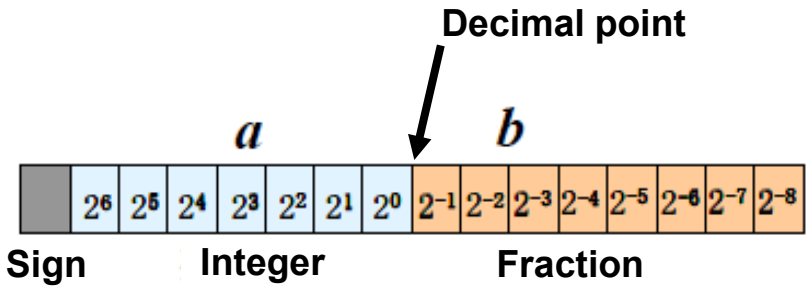
Q2. What is the principle that allows even a toy CPU to have five pipeline stages?
おもちゃのCPUでも5段パイプラインステージにできる原理は何か?

Q3. Can computers calculate 5 billion + 1 correctly?
コンピュータは,20億+1を正しく計算できるか?

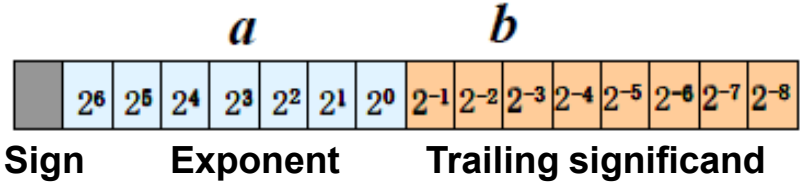
Q4. Your program is correct. But 100 times slower than colleague's program. What should you do?
あなたのプログラムは実行結果が正しい.でも同僚のプログラムより100倍遅い.気付くべき点は?

Q5. What can you do for the energy and digital deficit problem derived from depending on imported computers?
海外製計算基盤依存の結果生じたエネルギー・デジタル赤字問題の解決方法は何か?

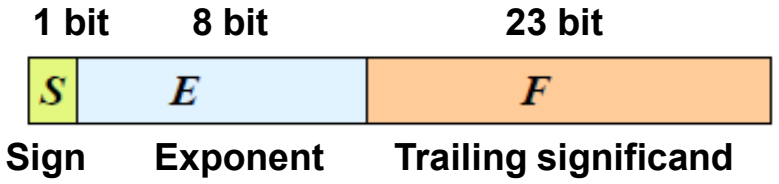
Fixed-point number vs Floating-point number



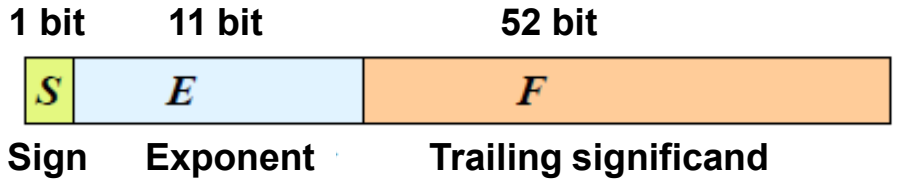
(a) Fixed-point format: a + b



(b) Floating-point format: b x 2^a



(a) Single precision



(b) Double precision

Binary16 and binary128 format

Table 3.5—Binary interchange format parameters

Parameter	binary16	binary32	binary64	binary128	binary{*k*} ($k \geq 128$)
k, storage width in bits	16	32	64	128	multiple of 32
p, precision in bits	11	24	53	113	$k - \text{round}(4 \times \log2(k)) + 13$
emax, maximum exponent *e*	15	127	1023	16383	$2^{(k-p-1)} - 1$
Encoding parameters					
bias, $E - e$	15	127	1023	16383	*emax*
sign bit	1	1	1	1	1
w, exponent field width in bits	5	8	11	15	$\text{round}(4 \times \log2(k)) - 13$
t, trailing significand field width in bits	10	23	52	112	$k - w - 1$
k, storage width in bits	16	32	64	128	$1 + w + t$

The function round() in Table 3.5 rounds to the nearest integer.

For example, binary256 would have $p = 237$ and $emax = 262143$.

Representable range and Accuracy

- C has no framework to detect integer-overflow
  - INT32_MAX + 1 ⇒ INT32_MIN with no report
  - Σint32 will not represent the population of the world

- Float can represent larger than 10billion, but …
  - INT32 … sign:1bit, integer:31bits
    - 2147479552 + 1 ⇒ 2147479553
  - FP64 … sign:1bit, exp:11bits, significand:52bits
    - 2147479552.0 + 1.0 ⇒ 2147479553.0
  - FP32 … sign:1bit, exp:8bit, significand:23bits
    - 2147479552.0 + 1.0 ⇒ 2147479552.0 (incorrect)

Goal of today

Q1. What are three main components of a computer?
コンピュータの主要構成要素を3つ挙げよ

Q2. What is the principle that allows even a toy CPU to have five pipeline stages?
おもちゃのCPUでも5段パイプラインステージにできる原理は何か?

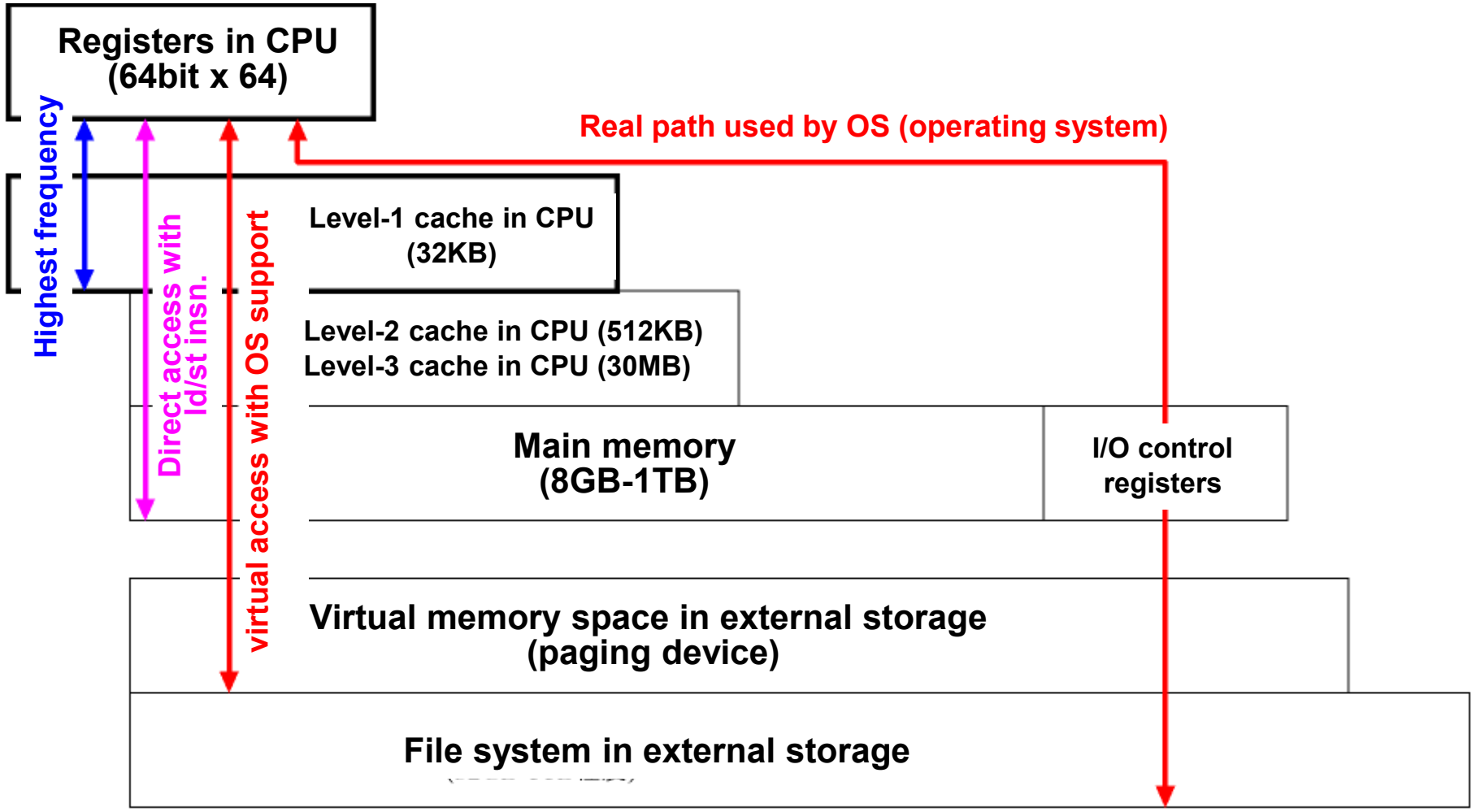
Q3. Can computers calculate 5 billion + 1 correctly?
コンピュータは,20億+1を正しく計算できるか?

Q4. Your program is correct. But 100 times slower than colleague's program. What should you do?
あなたのプログラムは実行結果が正しい.でも同僚のプログラムより100倍遅い.気付くべき点は?

Q5. What can you do for the energy and digital deficit problem derived from depending on imported computers?
海外製計算基盤依存の結果生じたエネルギー・デジタル赤字問題の解決方法は何か?

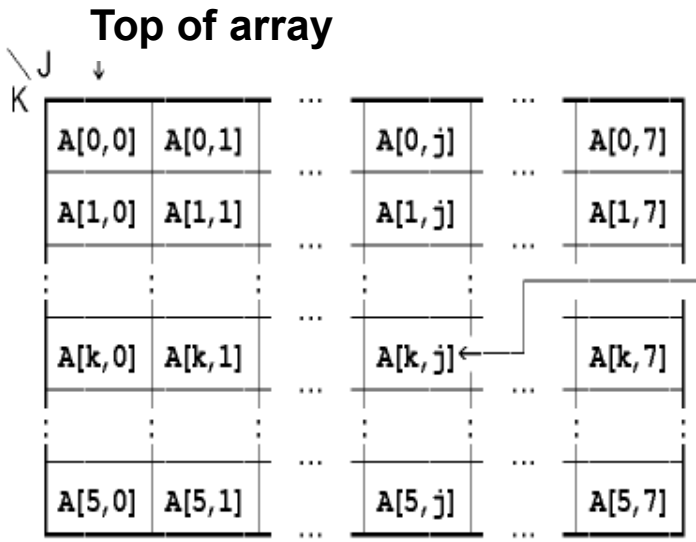
Memory Hierarchy

Too slow without cache memory



Alignment of data-structure and cache-structure (2D-array)

- **Location of data depends on programming language.**



Loc of A[k,j] is "A[0,0] + elem_size * (num_j * k + j)"
or "A[0,0] + elem_size * (num_k * j + k)"

In case of C:

```
for (i=0; i<1000; i++)   /* A[0][0] ... A[999][999]
  for (j=0; j<1000; j++)/* Inner-most loop should right-side index
    A[i][j] = xxxx;  /* A[0][0] A[0][1] … A[999][998] A[999][999]
```

In case of FORTRAN:

```
DO 10 I=1,1000
  DO 10 J=1,1000      ··· Inner-most loop should left-side index
    A(J,I) = xxxx     ··· A[1][1] A[2][1] … A[999][1000] A[1000][1000]
10 CONTINUE
```

Which one is faster?

```
double A[2048][2048];
double B[2048][2048];

main()
{
  int i, j, k;
  for (i=0; i<64; i++) {
    for (j=0; j<2048; j++) {
      for (k=0; k<2048; k++)
        A[k][j] = B[k][j]*2.0+1.0;
} } }
```

time: 10.94 sec

```
double A[2048][2048];
double B[2048][2048];

main()
{
  int i, j, k;
  for (i=0; i<64; i++) {
    for (j=0; j<2048; j++) {
      for (k=0; k<2048; k++)
        A[j][k] = B[j][k]*2.0+1.0;
} } }
```

time: 0.6 sec

Goal of today

Q1. What are three main components of a computer?
コンピュータの主要構成要素を3つ挙げよ

Q2. What is the principle that allows even a toy CPU to have five pipeline stages?
おもちゃのCPUでも5段パイプラインステージにできる原理は何か?

Q3. Can computers calculate 5 billion + 1 correctly?
コンピュータは,20億+1を正しく計算できるか?

Q4. Your program is correct. But 100 times slower than colleague's program. What should you do?
あなたのプログラムは実行結果が正しい.でも同僚のプログラムより100倍遅い.気付くべき点は?

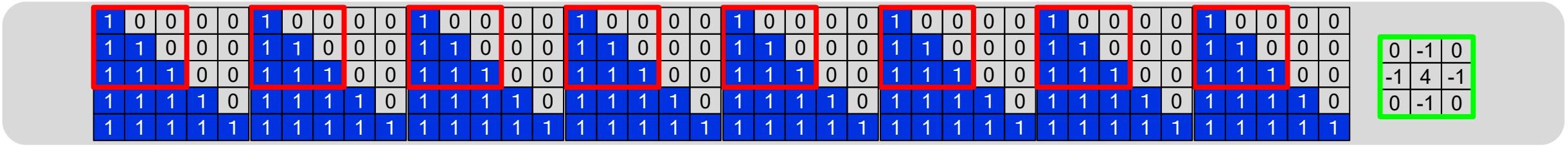
Q5. What can you do for the energy and digital deficit problem derived from depending on imported computers?
海外製計算基盤依存の結果生じたエネルギー・デジタル赤字問題の解決方法は何か?

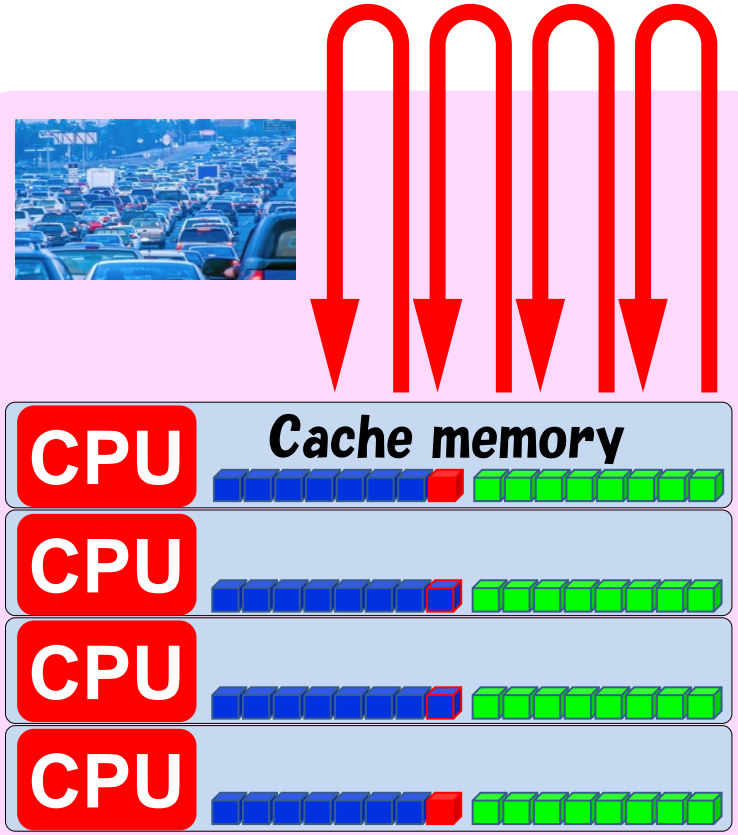
Energy consumption of servers

CO2: 0.5Mt/TWh

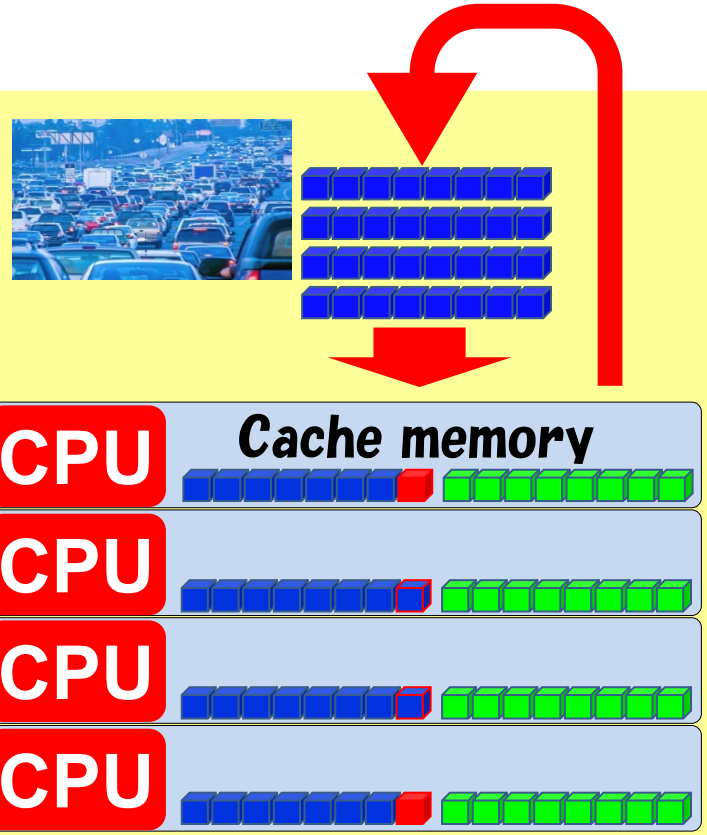
*Ref. LCS/JST https://www.jst.go.jp/lcs/pdf/fy2021-pp-01.pdf	Basic	AI	Total	CO2
*Consumed energy (2019 global)			67,000TWh	33.5Gt
*Consumed energy in servers (2018 global)	90TWh	15TWh	105TWh	0.05Gt
*Consumed energy in servers (2030) green computing	190TWh	320TWh	510TWh	0.25Gt
*Consumed energy in servers (2030) as is	450TWh	1,200TWh	1,700TWh	0.85Gt
*Consumed energy in servers (2050) green computing	3,400TWh	7,200TWh	11,000TWh	5.5Gt
*Consumed energy in servers (2050) as is	53,000TWh	221,000TWh	270,000TWh	135Gt
*Consumed energy in servers (2018 Japan)	6TWh	0.7TWh	7TWh	3.5Mt
*Consumed energy in servers (2030) green computing	13TWh	4TWh	17TWh	8.5Mt
*Consumed energy in servers (2030) as is	30TWh	16TWh	46TWh	23Mt
*Consumed energy in servers (2050) green computing	229TWh	97TWh	330TWh	165Mt
*Consumed energy in servers (2050) as is	3,500TWh	3,000TWh	6,500TWh	3250Mt

Architectural Innovation for Green Computing

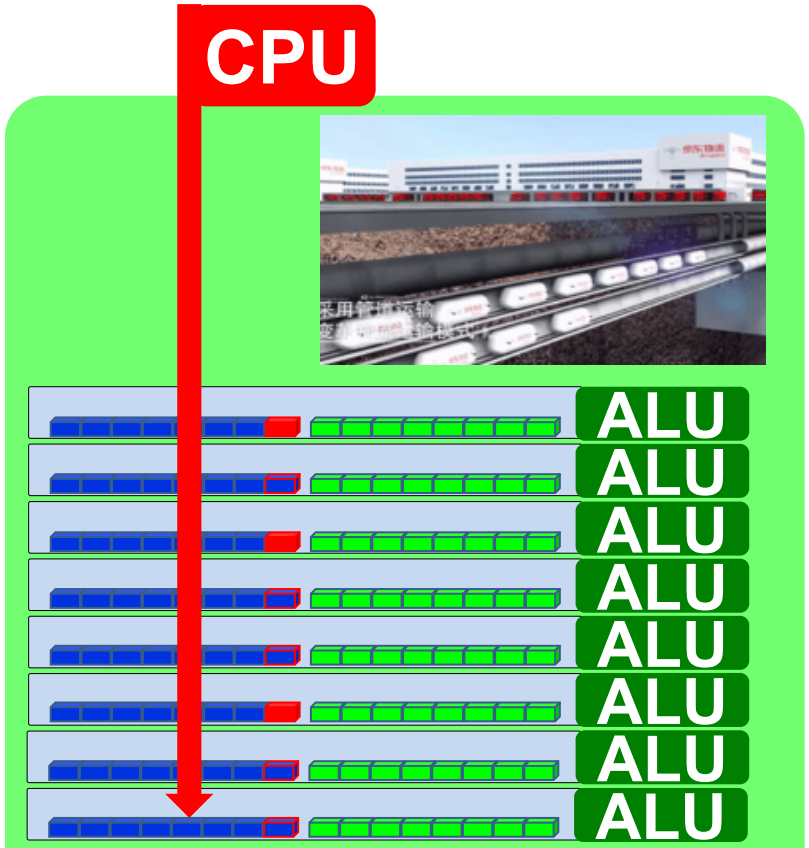




CPU: much congestion

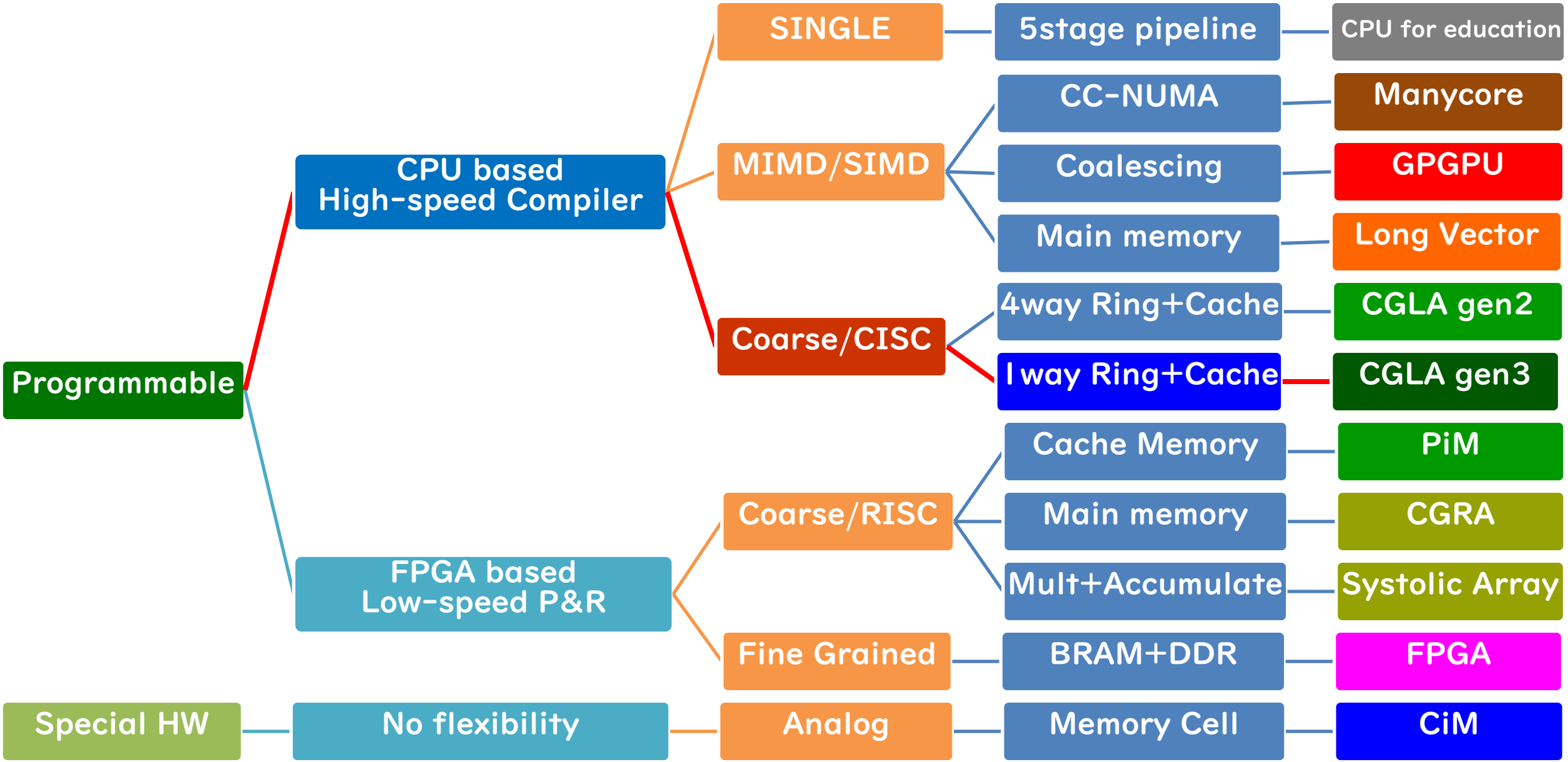


GPGPU: Coalescing



Green Computing

Modern Computing Platforms



Summary

- Q1. What are three main components of a computer?
  コンピュータの主要構成要素を3つ挙げよ
- Q2. What is the principle that allows even a toy CPU to have five pipeline stages?
  おもちゃのCPUでも5段パイプラインステージにできる原理は何か?
- Q2. Can computers calculate 5 billion + 1 correctly?
  コンピュータは,20億+1を正しく計算できるか?
- Q3. Your program is correct. But 100 times slower than colleague's program. What should you do?
  あなたのプログラムは実行結果が正しい.でも同僚のプログラムより100倍遅い.気付くべき点は?
- Q4. What can you do for the energy and digital deficit problem derived from depending on imported computers?
  海外製計算基盤依存の結果生じたエネルギー・デジタル赤字問題の解決方法は何か?