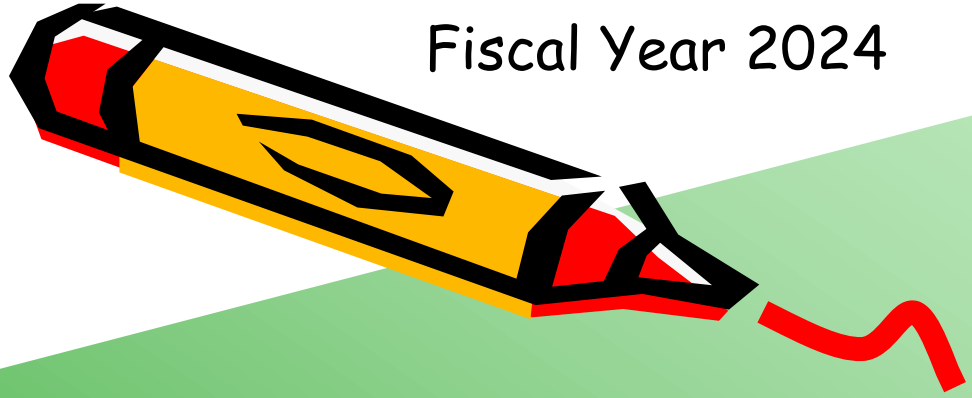


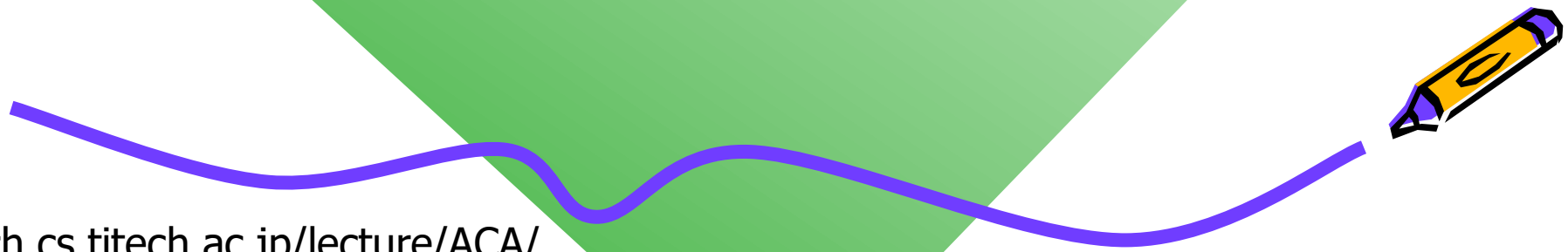
Fiscal Year 2024

Ver. 2025-01-30a

Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

Advanced Computer Architecture

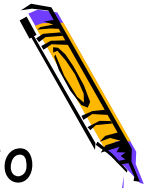
Final Report



www.arch.cs.titech.ac.jp/lecture/ACA/
Room No. W8E-308, Lecture (Face-to-face)
Mon 13:30-15:10, Thr 13:30-15:10

Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

Final report of Advanced Computer Architecture



1. Please submit your final report describing your answers to all questions in a PDF file
   via E-mail (kise [at] c.titech.ac.jp) by February 10, 2025
   - E-mail title should be "Report of Advanced Computer Architecture"
2. Please submit the report in 15 pages or less on A4 size PDF file, including the cover page.

3. You can discuss it with your colleague, but try to solve the questions yourself. Enjoy!

1. Academic paper reading

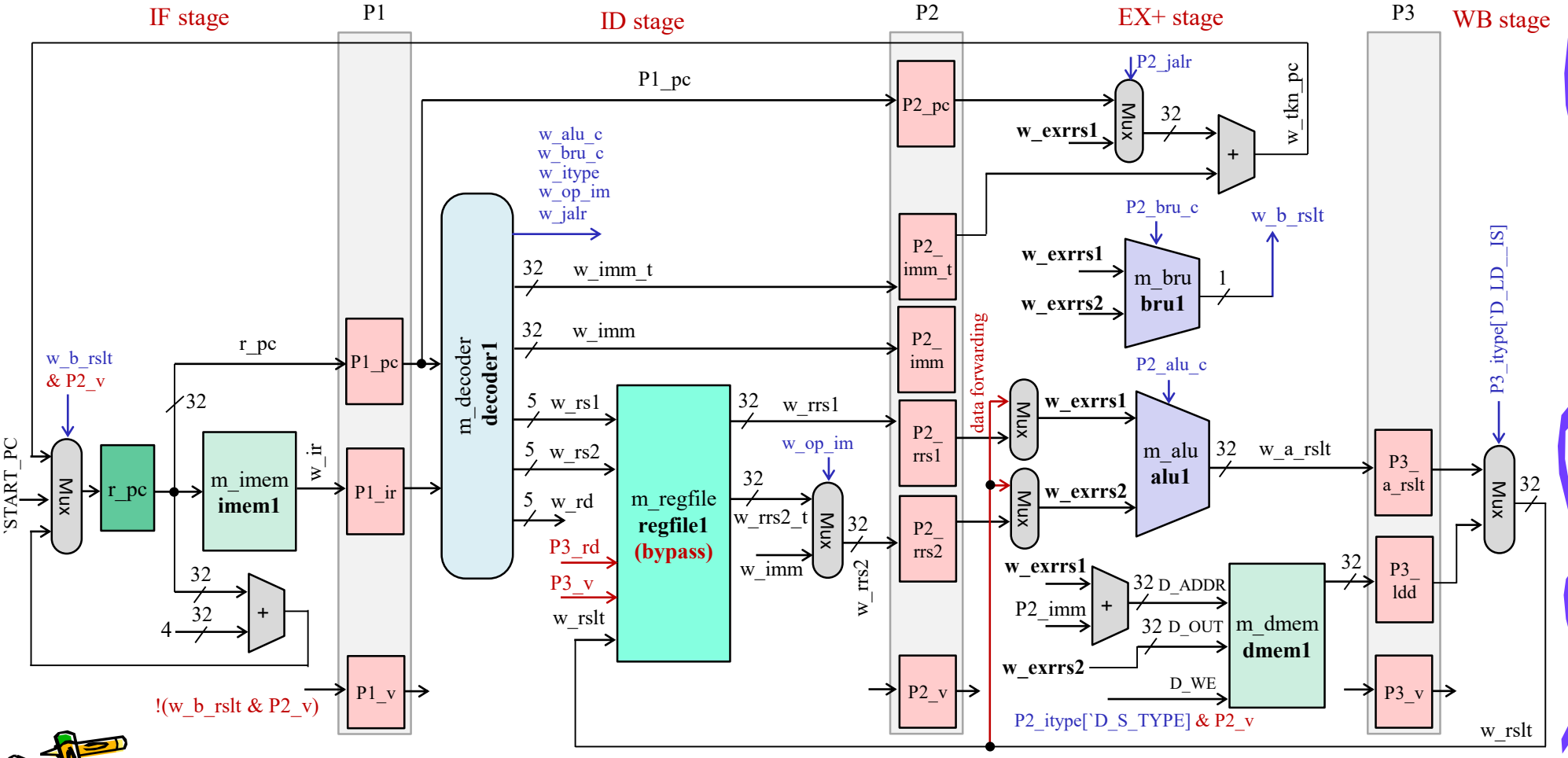
- Select an academic paper from the list below and
  - In your own word, describe the problem that the authors try to solve,
  - Describe the key idea of the proposal,
  - Describe your opinion why the authors could solve the problem although there may be many researchers try to solve similar problems.
  - Choose the figure you consider to be the most important in the paper and explain why.
- List
  - Increasing Processor Performance by Implementing Deeper Pipelines, ISCA,2002
  - Combining Branch Predictors, WRL TN, 1993
  - Prophet/critic hybrid branch prediction, ISCA, 2004
  - Focused Value Prediction, ISCA, 2020
  - Emulating Optimal Replacement with a Shepherd Cache, MICRO, 2008
  - Clockhands: Rename-free Instruction Set Architecture for Out-of-order Processors, MICRO, 2023

2. four-stage pipelining processor

- Describe a 4-stage pipelining scalar processor (rvcore_4s) in Verilog HDL. The report should include the description of module m_rvcore_4s.
- Verify the described code by compaing vefiry.txt generated by simulations of rvcore1 and rvcore_4s.

rvcore_4s : 4-stage pipelining processor with data forwarding

- The strategy is to separate the instruction fetch (IF) step, instruction decode (ID) step, and write back(WB) step, and other (EX, MA) steps. The first stage is named IF. The second stage is named ID. The third stage is named EX+. The last stage is named WB.

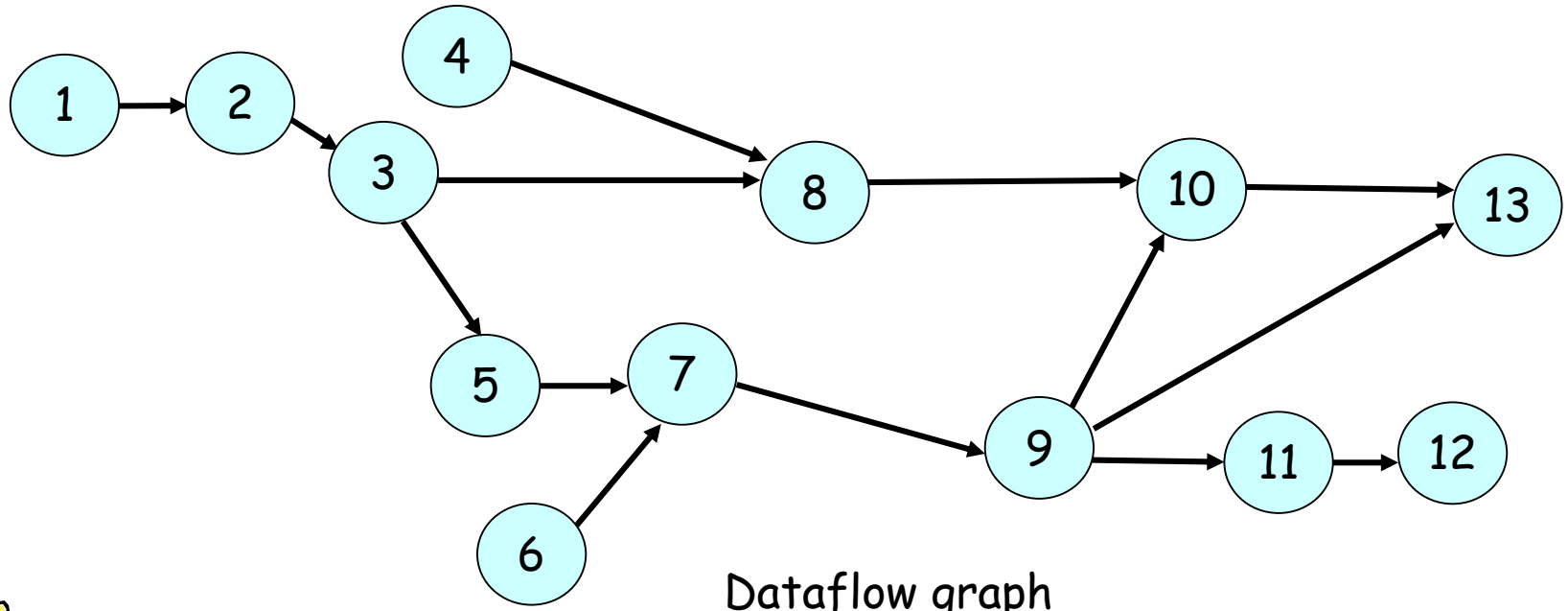


four-stage pipelining processor

- login the gateway server of ACRi room with your username
  - `ssh username@gw.acri.c.titech.ac.jp`
- login a compute server of ACRi room, select one among vs100, vs200, vs300, and vs400
  - `ssh vs300`
- copy the project directory to your working directory
  - `cd`
  - `mkdir -p aca`
  - `cd aca`
  - `cp -r /home/tu_kise/aca/rvcore_2s/ .`
  - `cp -r /home/tu_kise/aca/rvcore_3s/ .`
- simulate and test two-stage pipelining processor
  - `cd rvcore_2s`
  - `make`
  - `make run`
- implement your four-stage pipelining processor, simulate it, and vefity it
  - `cd ~/aca/rvcore_4s`
  - `emacs proc1.v` (please use your favorit text editor)
  - `make`
  - `make run`
  - `diff verify.txt ../rvcore_2s/verify.txt`

3. OoO execution and dynamic scheduling

- Draw the cycle by cycle processing behavior of these 13 instructions
- Modify this dataflow graph by removing two edges of the graph so that the number of execution cycles is reduced. Draw another cycle by cycle processing behavior of the modified graph.



Dataflow graph


Cycle 1
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 2
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 3
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 4
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 5
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 6
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 7
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 8
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 9
Instruction window
Issue
Execute
Commit
Retire
ROB
Cycle 10
Instruction window
Issue
Execute
Commit
Retire
ROB

4. Parallel programming

- Adjust the number of elements N so that this sequential program (main8.c) takes about 5 second. Use this adjusted value for N. Use the `time` command to measure the execution time.
  Please use –O0 optimization flag while compiling.
- Describe an efficient parallel program for the sequential program of main8.c using LOCK, UNLOCK, and BARRIER of pthread assuming a shared memory architecture of 4 cores.
- Explain why your code runs correctly and why your code is efficient.
- Show your speedup over the sequential execution. To measure the execution time, use a computer with four or more cores.

main8.c

```
#include <stdio.h>
#include <math.h>
#define N   30   /* the number of grids */
#define TOL 15.0 /* tolerance parameter */
float A[N+2], B[N+2];
float diff;

void solve () {
    int i, done = 0;
    while (!done) {
        diff = 0;
        for (i=1; i<=N; i++) {
            B[i] = 0.333 * (A[i-1] + A[i] + A[i+1]);
            diff = diff + fabsf(B[i] - A[i]);
        }
        if (diff <TOL) done = 1;
//        for (i=0; i<=N+1; i++) printf(" %6.2f", A[i]);
//        printf("¥n");
        for (i=1; i<=N; i++) A[i] = B[i];
  }
}

int main() {
    int i;
    for (i=1; i<=N; i++) A[i] = 100+i*2;
    A[0] = 90;
    A[N+1] = 50+N*2;

    solve();
    for (i=N/2-10; i<=N/2; i++) printf(" %6.3f", B[i]);
    printf("¥n diff=%6.3f¥n", diff);
}
```

5. Building blocks for synchronization

- Implement your BARRIER() using some global variables, pthred lock, and unlock.
- Show your code and explain why your code runs correctly and why your code is efficient.

- Replace the barrier in the program in Question 4 with your designed one and measure the speedup over the sequential program.

6. Cache coherence protocols

- Select your favorite commercial Intel or AMD multi-core processor with more than three cores shipped after 2022
  - Describe the memory organization, including caches and main memory
    - cache line size, write policy, write allocate/no-allocate, direct-mapped/set-associative, the number of caches (L1, L2, and L3?)
  - Describe the cache coherence protocol used there