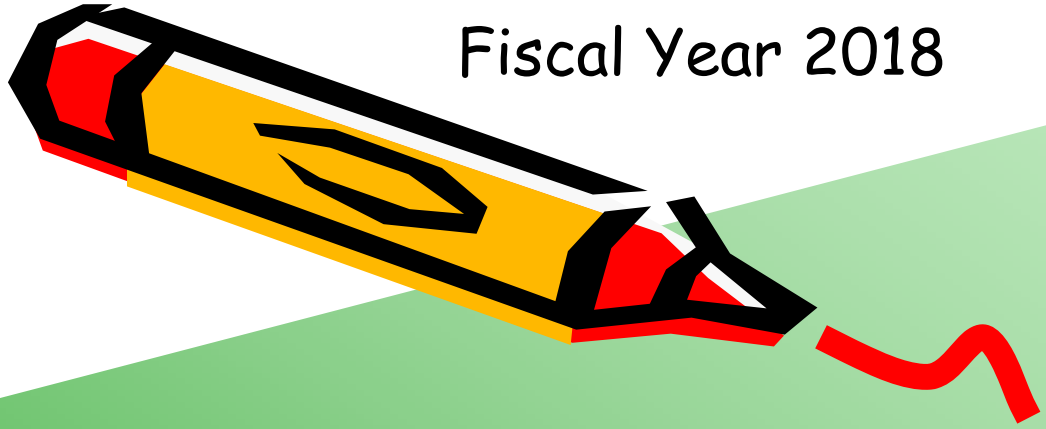


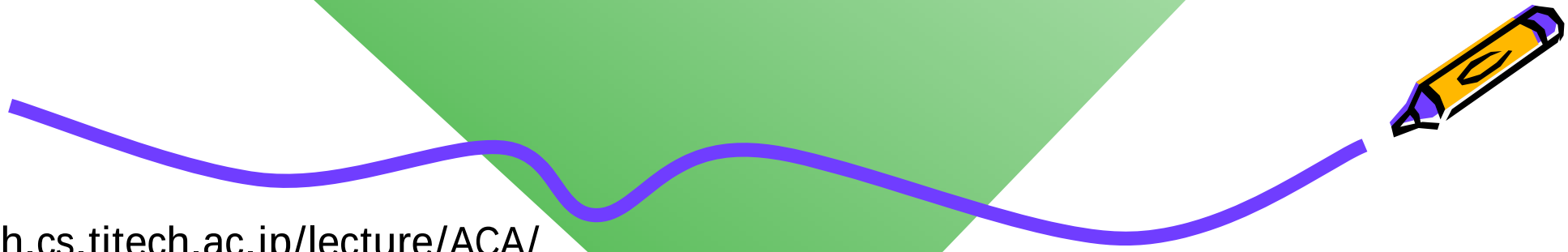
Fiscal Year 2018

Ver. 2019-02-01a

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

Advanced Computer Architecture

15. Final report



www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W936
Mon 13:20-14:50, Thr 13:20-14:50

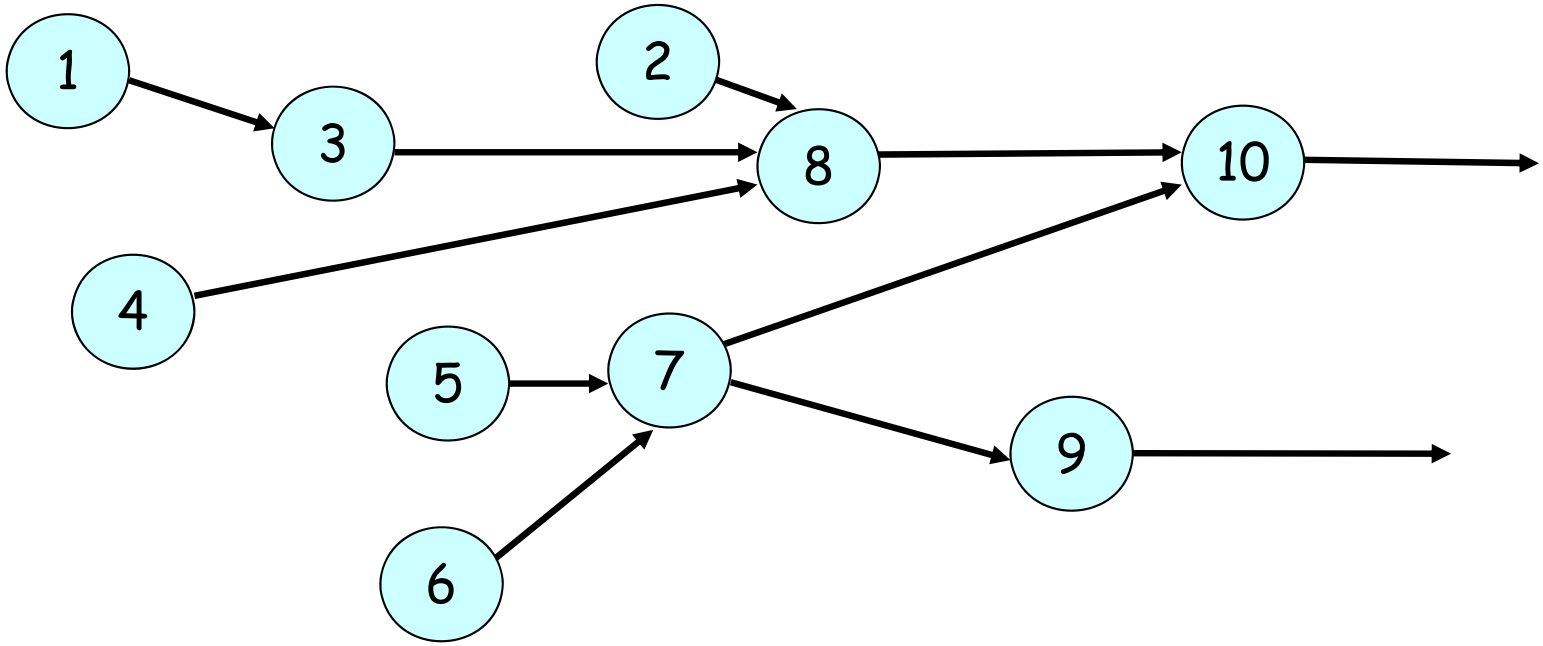
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

Final report

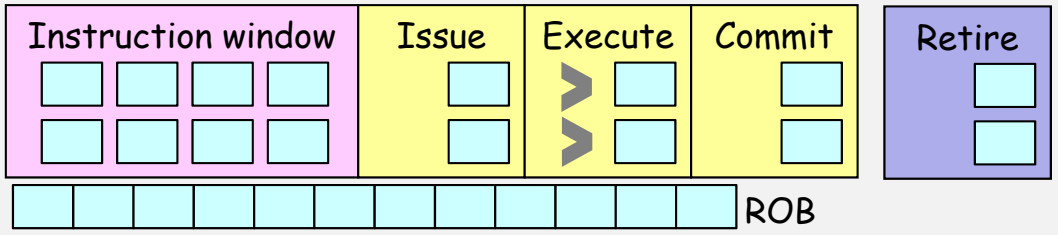
1. Submit your final report in a PDF file via E-mail by February 17, 2019
2. Enjoy!

1. OoO execution and dynamic scheduling

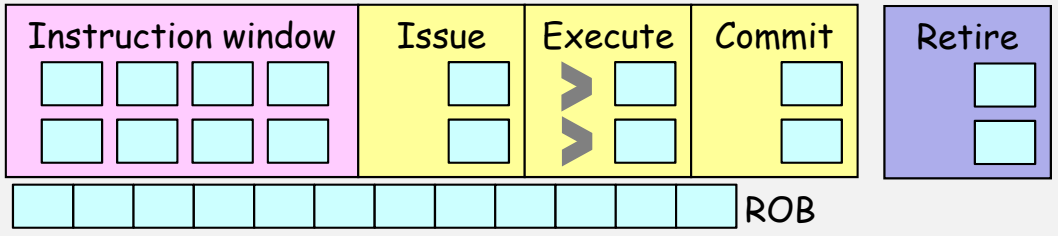
- Draw the cycle by cycle processing behavior of these 10 instructions
- Modify this dataflow graph and draw another cycle by cycle processing behavior of the graph having 10 instructions



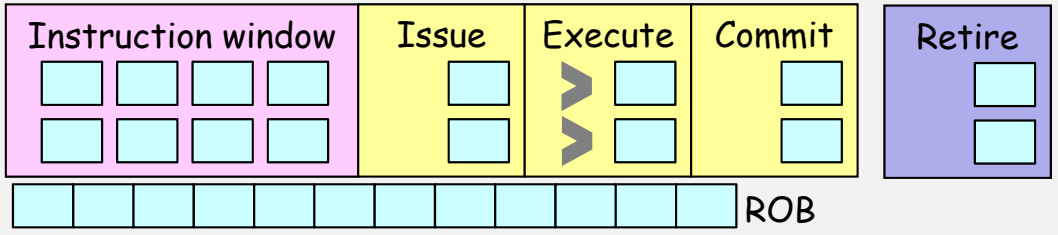
Cycle 1



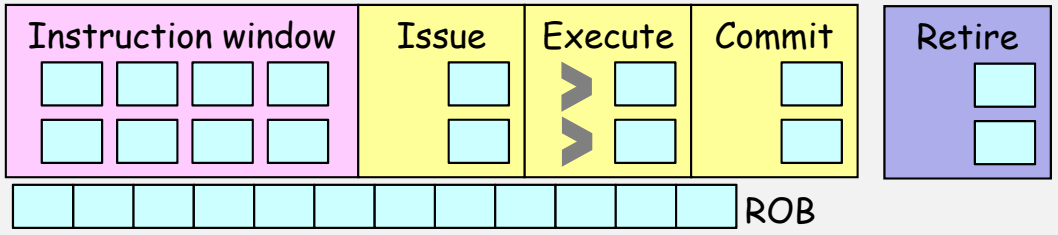
Cycle 2



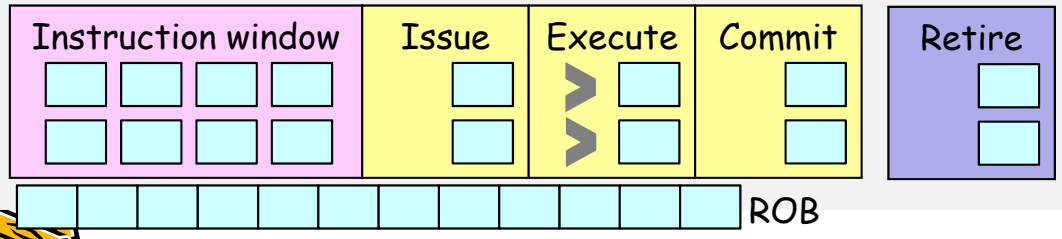
Cycle 3



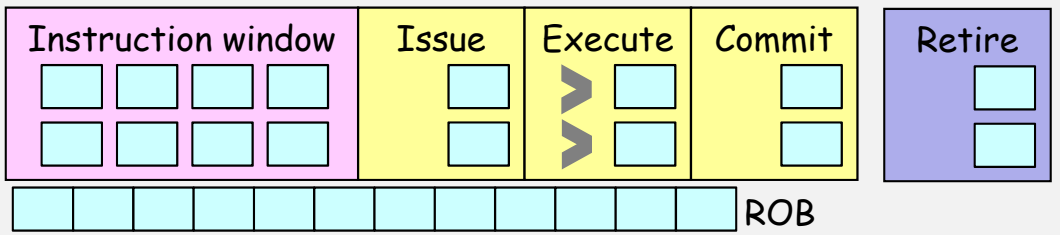
Cycle 4



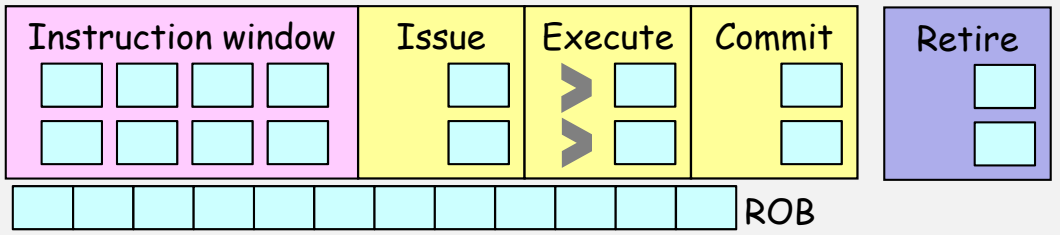
Cycle 5



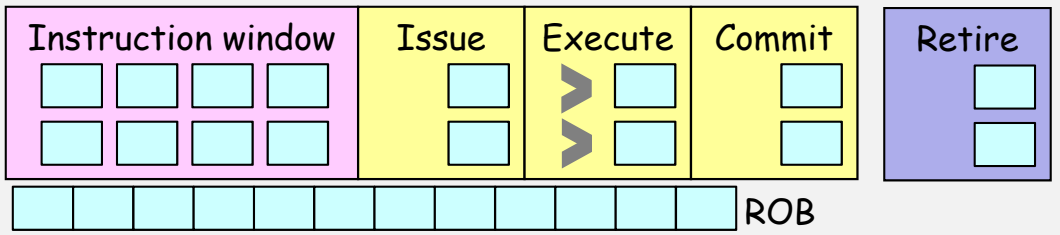
Cycle 6



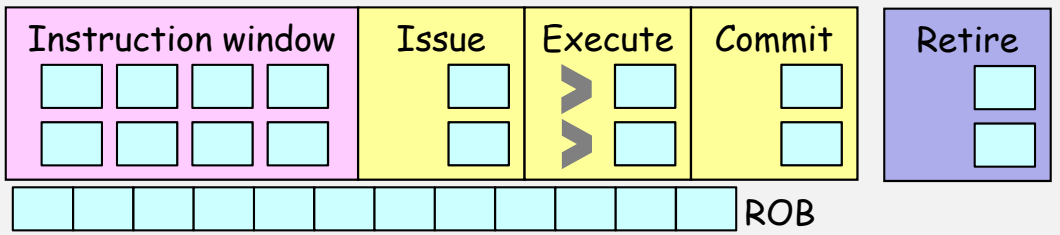
Cycle 7



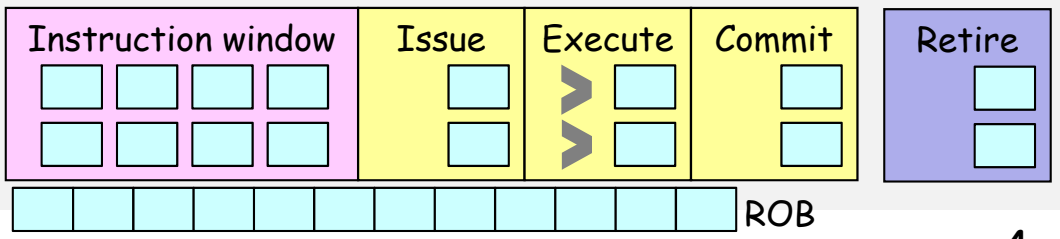
Cycle 8

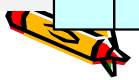


Cycle 9



Cycle 10





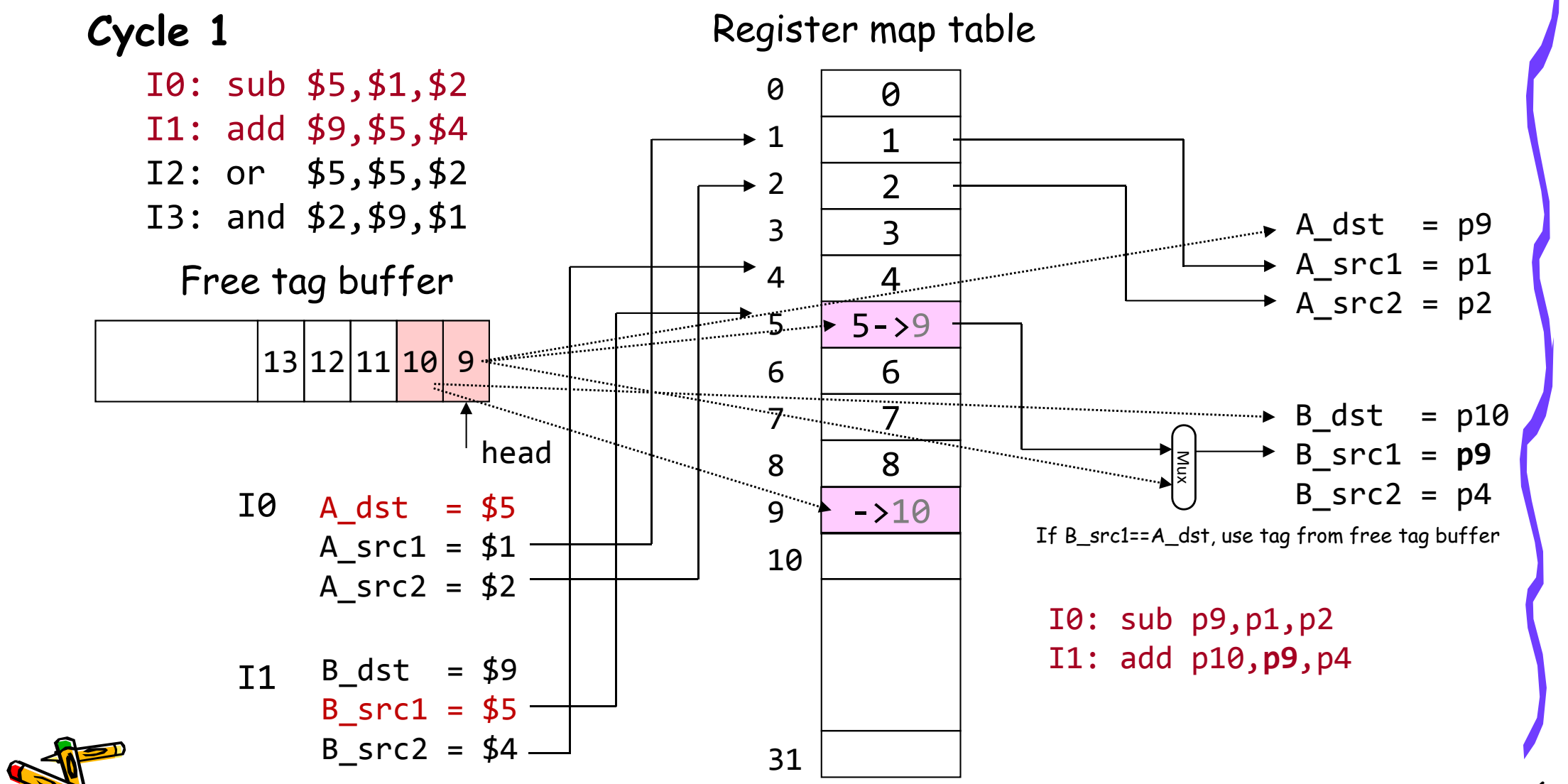
2. Register renaming of multi-instructions per cycle

1. Design a register renaming unit which renames two instructions per cycle in Verilog HDL. Please download rename01.v and rename02.v from the support page and refer them.
2. Design a register renaming unit which renames three instructions per cycle in Verilog HDL.

- Please modify a module RENAME in rename02.v referring the design which renames one instruction per cycle in rename01.v.
- The renamed instruction sequences by rename01.v and rename02.v must be the same.
- The report should include a block diagram, a source code in Verilog HDL, and obtained waveforms of your design.

Renaming two instructions per cycle for superscalar

- Renaming instruction I0 and I1



3. Parallel programming (The free lunch is over)

- Describe an efficient parallel program for the following sequential program using LOCK(), UNLOCK() and BARRIER().
- Explain why your cord runs correctly and why your code is efficient.

```
#define N 8      /* the number of grids */
#define TOL 15.0 /* tolerance parameter */
float A[N+2], B[N+2];

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

int main() {
    int i;
    for (i=1; i<N-1; i++) A[i] = B[i] = 100+i*i;
    solve();
}
```

main02.c

4. Building blocks for synchronization

- Fetch-and-increment reads an original value from memory and increments (adds one to) it in memory atomically
- Implement fetch-and-increment (FAI) using the load-linked/store-conditional instruction pair
  - Refer the discussion of implementing an atomic exchange EXCH
- Implement BARRIER() using FAI

5. Cache coherence protocols

- Select your favorite commercial multi-core processor
  - 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

6. Academic paper reading

- Select a paper from the list below and
  - 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.
- List
  - Emulating Optimal Replacement with a Shepherd Cache, MICRO-40, 2008
  - The V-Way Cache: Demand Based Associativity via Global Replacement, ISCA'05, 2005
  - Skewed Compressed Caches, MICRO-47, 2014
  - Prophet/critic hybrid branch prediction, ISCA'04, 2004
  - A new case for the TAGE branch predictor, MICRO-44, 2011