No Instruction Computing Using Pointers and Operations in Registers for Adaptable Architecture

IXPUG Annual Meeting 2020

Dr. Nagi Mekhiel
Department of Electrical, Computer and Biomedical Engineering
Ryerson University, Toronto, CANADA
nmekhiel@ee.ryerson.ca

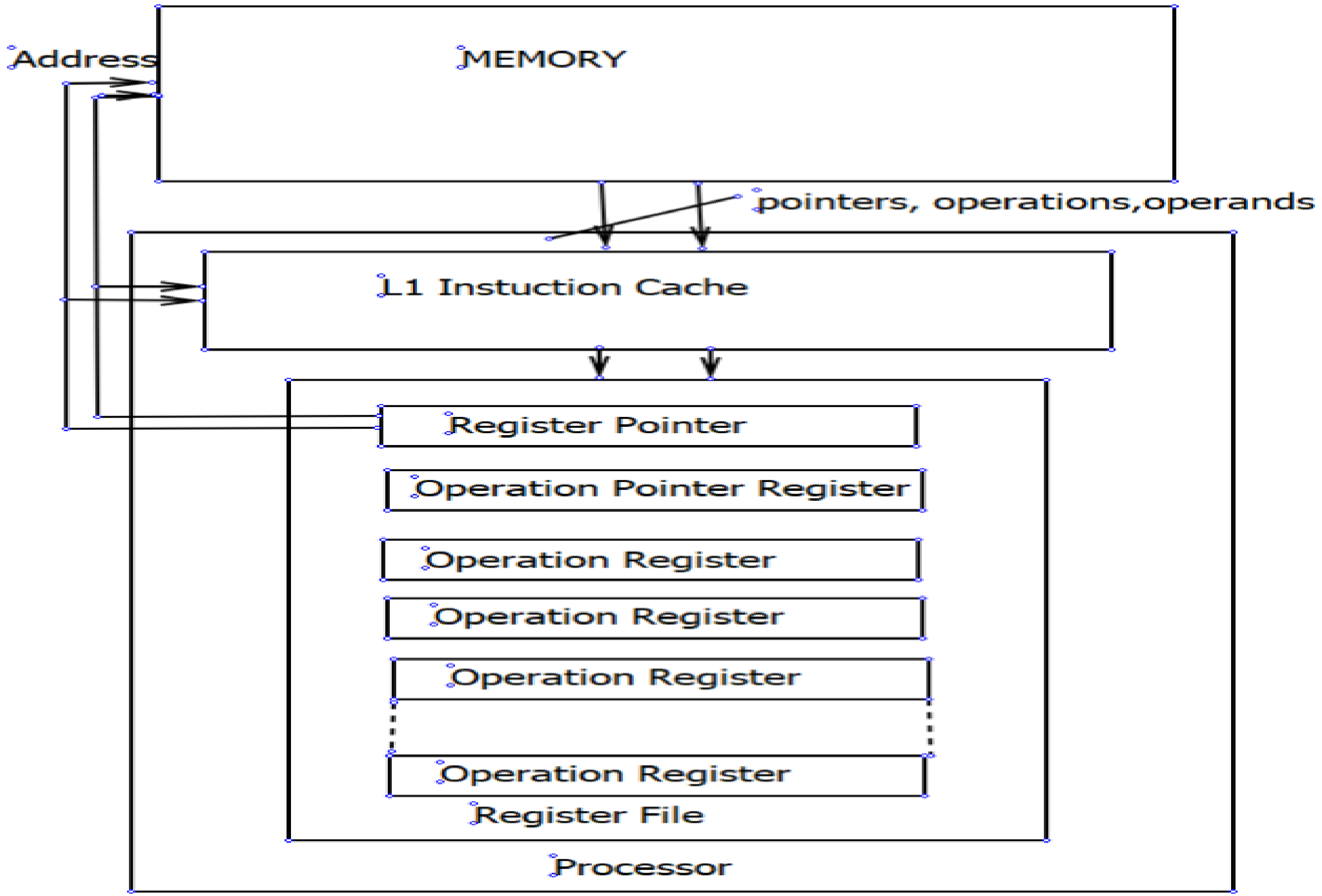
Introduction

- Current architectures depend on delivering maximum numbers of instructions per cycle
- Performance more dependent on memory bandwidth
- Must use program counter each cycle to fetch instructions from memory
- Issuing parallel instructions is complicated
- Operations have more localities than data and could move to registers

No Instructions Computing Concept

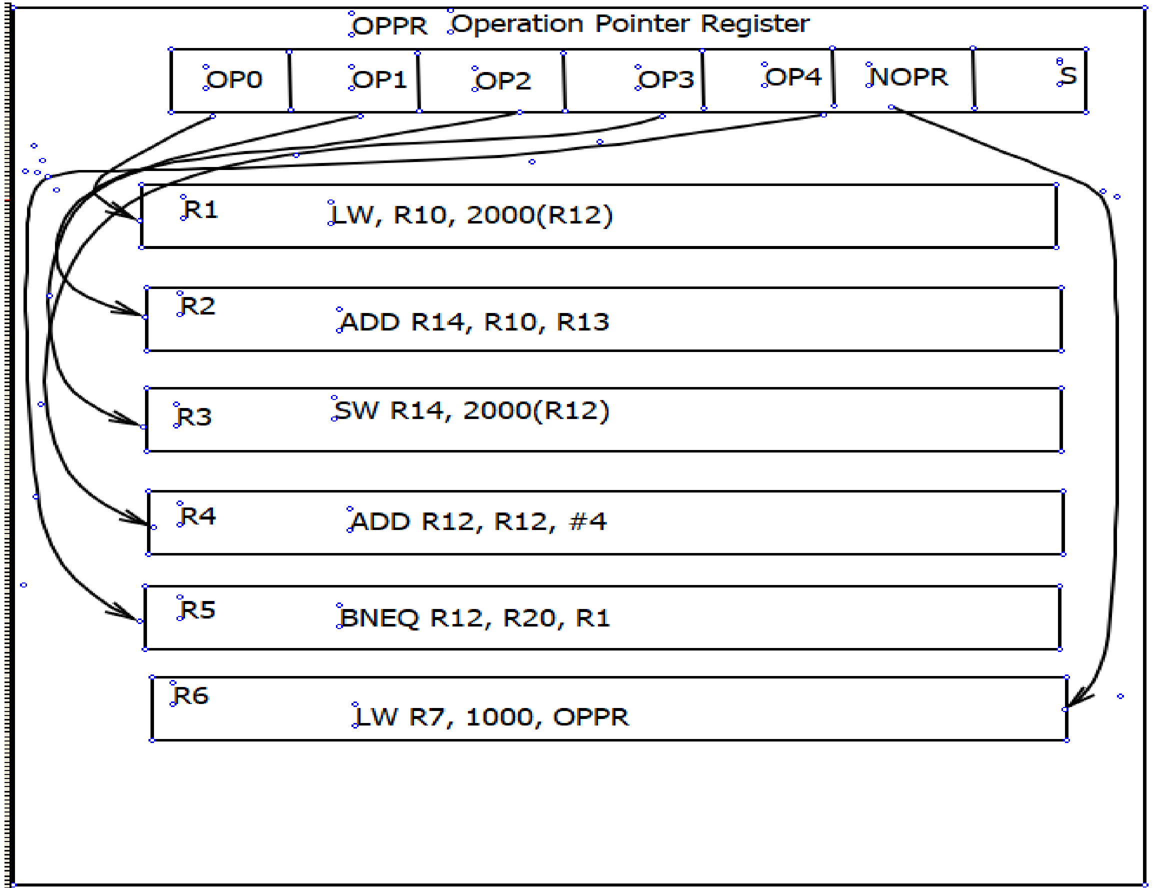
- It uses load to transfer operations and operands from memory to the register file
- It eliminates the use of a program counter that points only to one instruction at a time
- Single operation register pointer stores multiple pointers each points to one operation in another lower level registers
- The last field of register pointer points to the next operation register that loads the next block of code to be executed
- It executes multiple operations in parallel if the mode in operation pointer indicates a parallel operation

The Concept and Organization



Example of an application

- for(i=0; i<N;i++){A[i]=A[i]+B;}.



Performance Evaluation

- for(i=0; i<N;i++){A[i]=A[i]+B;}. Register access=1 cycle, Memory access = 2 cycle

Conventional Architecture

Total time = 5Nx1 cycle for executing instructions + 2Nx2 cycles for data + 5Nx2 cycle for instructions memory= 19N cycles

Memory BW requirements = (2N+5N)x4B/19N= 1.5B/cycle

No Instructions Computing

Total time= 5Nx1 cycle for executing instructions + 6x2 cycles to load the pointer and operations + 2Nx2 cycles for data load and store= (9N + 12) cycles

Memory BW requirements= 2Nx4B/(9N+12) = .89 B/cycle

Gain:

Performance gain= 19N/(9N+12) = about 200% and Memory BW reduction = 1.5B/.89 B= 168 %

Conclusions

- No Instructions Computing eliminates the use of program counter and reduces memory bandwidth requirements
- It is adaptable and supports SIMD and parallel operations Parallel operations from a single operation register pointer eliminates the overhead of synchronization
- Future work needs to implement it using suitable technology like FPGA and develop software tools to run actual applications

References

1-Nagi Mekhiel, Instructions with Indirect Register Indexing for Multiple and Exponential Operations to Improve Performance?, The 12thIEEE International Conference on Computer Engineering and Systems (ICCES 2018), Dec 18-19 2018, Cairo, Egypt

2-Nagi Mekhiel, Take the ‘No Instruction’ road past memory straits, EE Times Aug 23, 1999