International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 6, Issue 11, November-2017, Impact Factor: 3.578

A study of Microprocessor-Future Technologies

Kamlesh

Assistant Professor,
Baba Farid College, Bathinda (India)

ABSTRACT

Current microprocessors utilise the instruction-level parallelism by a deep processor pipeline and the superscalar instruction issue technique. VLSI technology offers several solutions for aggressive exploitation of the instruction-level parallelism in future generations of microprocessors. Technological advances will replace the gate delay by on-chip wire delay as the main obstacle to increase the chip complexity and cycle rate. The implication for the microarchitecture is that functionally partitioned designs with strict nearest neighbour connections must be developed. Among the major problems facing the microprocessor designers is the application of even higher degree of speculation in combination with functional partitioning of the processor, which prepares the way for exceeding the classical dataflow limit imposed by data dependences. In this paper we survey the current approaches to solving this problem, in particular we analyse several new research directions whose solutions are based on the complex uniprocessor architecture. A uniprocessor chip features a very aggressive superscalar design combined with a trace cache and superspeculative techniques. Superspeculative techniques exceed the classical dataflow limit where even with unlimited machine resources a program cannot execute any faster than the execution of the longest dependence chain introduced by the program's data dependences. Superspeculative processors also speculate about control dependences. The trace cache stores the dynamic instruction traces contiguously and fetches instructions from the trace cache rather than from the instruction cache. Since a dynamic trace of instructions may contain multiple taken branches, there is no need to fetch from multiple targets, as would be necessary when predicting multiple branches and fetching 16 or 32 instructions from the instruction cache. Multiscalar and trace processors define several processing cores that speculatively execute different parts of a sequential program in parallel. Multiscalar processors use a compiler to partition the program segments, whereas a trace processor uses a trace cache to generate dynamically trace segments for the processing cores. A datascalar processor runs the same sequential program redundantly on several processing elements where each processing element has different data set. This paper discusses and compares the performance potential of these complex uniprocessors.

Keywords: Advanced superscalar processor; Superspeculative processor; Trace processor; Multiscalar processor; Datascalar processor.

I. INTRODUCTION

Microprocessor is a electronic chip, that functions as the central processing unit of a computer.
For example: Washing machines, microwave ovens, mobile phones etc.
Its advance applications are Radar, Satellites, flights.
All processors are use the basic concept of stored program execution. program or instructions are stored sequentially in the memory. Every microprocessor has its own associated set of instructions. Instruction set for microprocessor is in two forms one in mnemonic, which is comparatively easy to understand and the other is binary machine code.

II. OBJECTIVES

1. To study know about Microprocessor 8086
2. To study know about future Technologies of Microprocessor.

III. REVIEW OF LITERATURE

Technology Trends In the past 20 years, microprocessors technology has experienced improvements in circuit integration and microprocessor throughput. The technology has grown rapidly due to transistor speed, energy scaling and core micro architecture advances powered by Moore's law. In every generation (two years), transistor density has doubled as their dimensions have been reduced by 30% (shrinking their area 50%), and circuits have become 40% faster increasing the whole system performance. However, due to battery capacity and chip reliability (heat dissipation limits), power consumption has been the key limiting factor for performance scaling in the single-core microprocessor technology. In the past decade, multi-core microprocessors have become the major design trend. Limits in instruction level parallelism (ILP) and power dissipation constraints have triggered the high performance microprocessor roadmap to enter the multi-core era, starting from the high-end server processors and moving to the low-end hand-held mobile device processors. A multi-core micro architecture provided an effective alternative to improve throughput performance of parallel programs while keeping power consumption under the control. To improve efficiency, single-thread performance was sacrificed and instead multiple cores were joined on a single chip when more transistors became available. The more threads accommodated in the application set, the more efficient the processors became. Recently the typical pattern among multi-core CPU products is to keep the number of cores constant within a generation and double the number of transistors within each core . By exploiting Moore’s Law to replicate cores, multi-core architectures increased computational performance. However, there is no real benefit if the software has no parallelism . 2108 2013 Proceedings of PICMET '13: Technology Management for Emerging Technologies. Core micro architecture techniques took advantage of the abundance of transistor integrity to deliver improved performance; nevertheless growing power densities are still the major constraint to performance improvements. Initially, multi-core processors were designed with a step back in corelevel frequency allowing throughput increase at affordable power; however the power consumption and dissipation problem did not disappear with the multi-core era . With a flat power budget, from mobile platforms to PCs and workstations to the largest supercomputers being all power limited, power efficiency is one of the primary metrics for, and driver of, microprocessor designs. Power and heat management are the two major concerns that are more pronounced with the addition of multiple cores. As power continues to limit performance scaling, researchers forecast that processor designs will be forced to use large-scale parallelism and heterogeneous cores (application-customized), or a few large cores and a large number of small cores operating at low frequency and low voltage, as alternatives to achieve performance and energy efficiency. B. Forecasting Tools Technology forecasting is the act of forecasting inventions, innovations, or diffusion of technologies. It is a procedure of collecting data and analyzing them to predict future technological developments and its social effects . It is a popular technique among companies because it can be used to design future products to outperform competitors. Specifically, technology forecasting provides companies with a capability of studying the impact of past products and comparing them with the new product; which leads to a better understanding of the position of new technology. Conventional technology forecasting methods rely on techniques based on complex mathematics and/or expert judgment, and, can be classified under three categories - Time Series, Judgmental, and Causal/Econometric Method.

VI. LIMITATIONS

• Single variable based prediction: Technology is impacted by different attributes; it is hard to find the sole characteristic/variable that will impact technology in future
• Preference changes over time is not considered, thereby, unsuited for dynamic trade-offs
• Correlation between technology attributes is not considered. Technology attributes are assumed independent; for less known technologies it is difficult to detach the attributes.
• Lack of a multiple output model. Current methods work with a single output at a time; the outputs are fixed and there is no ability to waive any of them. Technology forecasting using Data Envelopment Analysis (TFDEA) is recognized as a powerful forecasting method in literature that addresses the above gaps. It is a non-parametric method that can incorporate multiple inputs and outputs to identify the best performers at each observation period and forecast the technology trend accordingly. It does not require a mathematical specification of functional relations between inputs and outputs. Technology forecasting via DEA is however very sensitive to the choice of variables. Therefore the inputs and outputs parameters need to be carefully selected.

V. RESEARCH METHODOLOGY

The study is based on secondary sources of data/ information.

- Different books,
- Journals,
- Newspapers and
- Relevant websites

Microprocessor 8085
4 Pin Diagram of 8085

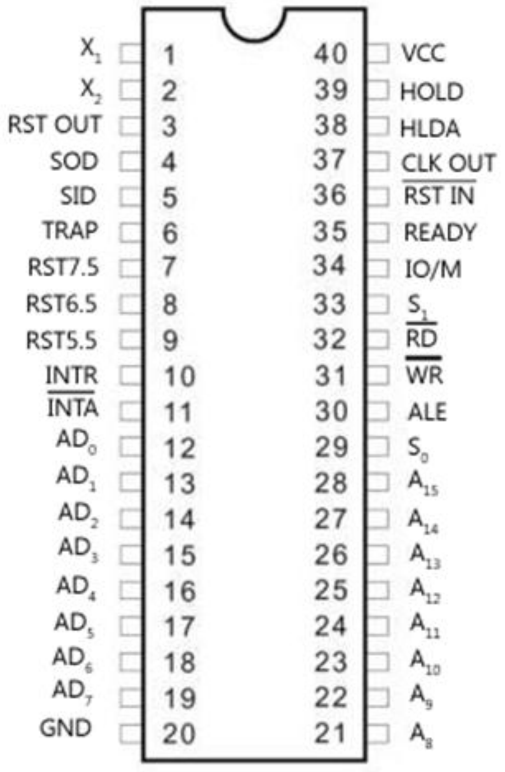


Fig: pin diagram of 8085

8085 up is an 8-bit general purpose microprocessor capable of addressing 64Kb of memory.
The device has 40 pins,+5 V power supply, operate on 3 to 5 MHZ frequency single phase clock.
All the signals can be classified in 8085 up pin diagram into six groups –

1**)** **Address Bus**: in this 16 signals lines. These lines are splits into two segments **-**
a) A_{15}-A_8 – unidirectional and used for the higher order address (MSB).
b)AD_7-AD_0 – Dual purpose such as data bus as well as lower address data bus(LSB).

2) Control and status signals: these signals are used to identify the nature of operation.
Three control signals that are-
RD – it is a active low signal. Which indicate that the selected IO or Memory device is to be read and data is available on the data bus.
WR-it is a active low signal which indicate that the data on the data bus are to be written into a selected memory or IO location.
ALE- it is a +ve going pulse generated everytime the 8085 begins an operation (machine cycle): which indicate that the bits on AD_7-$AD_0$are address bits.
Three status signals that are –
IO/M- this is a status signal used to differentiate between IO and Memory operations.when it is hign then IO operation and When it is low then Memory operation.
S1 and S0- status signals,similar to IO/M,can identify various operations.that are rarely used in the systems.

3) Power supply:
VCC : +5 V
VSS : Ground
4) Clock Frequency:
X1, X2**:** A crystal (RC,LC N/W) is connected at these two pins. this frequency is internally divided by 2.
CLK OUT: clock output this signal can be used as the system clock for the other devices.

5) Externally initiated signals: In this
Five interrupt signals: TRAP, RST 7.5, RST 6.5, RST 5.5 ,INTR.
INTA: interrupt acknowledge
RESET IN: It is a active low signal. When active program counter is set to zero.
RESET OUT: This signal indicates that the MPU is being reset, the signal can be used to reset other devices.
READY: If ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If ready is low, the CPU will wait for ready to go high before completing the read or write cycle.
HOLD: this signal indicate that another master is requesting the use of the address and data buses.
HLDA: HOLD Acknowledge indicates that the CPU has received the Hold request and that it will relinquish the buses I the next clock cycle. HLDA goes low after the HOLD request is removed. The CPU takes the buses one half clock cycle HLDA goes low.

6) Serial I/O ports:
SOD: serial output data line. The output SOD is set or reset as specified by the SIM instruction.
SID: Serial input data line, the data on this line is loaded into accumulator whenever a RIM instruction is executed.
In this data bits are sent over a single line one bit at a time.
For ex: Transmission over phone lines.

7) Instruction Set
An instruction is a binary pattern designed inside a microprocessor to perform a specific function. Each instruction is represented by 8 bit binary value.
Types of instruction set:

1) Data transfer instructions:
Instructions, which are used to transfer data from one register to another register, from memory to register or register to memory, come under this group. Examples are: MOV, MVI, LXI, LDA, STA etc. When an instruction of data transfer group is executed, data is transferred from the source to the destination without altering the contents of the source. For example, when MOV A, B is executed the content of the register B is copied into the register A, and the content of register B remains unaltered. Similarly, when LDA 2500 is executed the content of the memory location 2500 is loaded into the accumulator. But the content of the memory location 2500 remains unaltered.
EXAMPLES:
1. MOV r1, r2 (Move Data; Move the content of the one register to another). [r1] <-- [r2]
2. MOV r, m (Move the content of memory register). r <-- [M]
3. MOV M, r. (Move the content of register to memory). M <-- [r]
4. MVI r, data. (Move immediate data to register). [r] <-- data.
5. MVI M, data. (Move immediate data to memory). M <-- data.
6. LXI rp, data 16. (Load register pair immediate). [rp] <-- data 16 bits, [rh] <-- 8 LSBs of data.
7. LDA addr. (Load Accumulator direct). [A] <-- [addr].
8. STA addr. (Store accumulator direct). [addr] <-- [A].
9. LHLD addr. (Load H-L pair direct). [L] <-- [addr], [H] <-- [addr+1].
10.SHLD addr. (Store H-L pair direct) [addr] <-- [L], [addr+1] <-- [H].
11.LDAX rp. (LOAD accumulator indirect) [A] <-- [[rp]]
12.STAX rp. (Store accumulator indirect) [[rp]] <-- [A].
13.XCHG. (Exchange the contents of H-L with D-E pair) [H-L] <--> [D-E].
7 Instruction Set contd....

2) Arithmatic instructions:

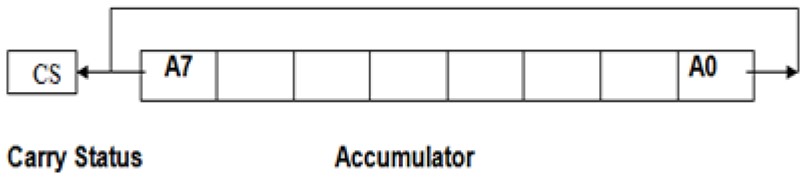
The instructions of this group perform arithmetic operations such as addition, subtraction; increment or decrement of the content of a register or memory.
Examples:
1). ADD r. (Add register to accumulator) [A] <-- [A] + [r].
2) .ADD M. (Add memory to accumulator) [A] <-- [A] + [[H-L]].
3).ADC r. (Add register with carry to accumulator). [A] <-- [A] + [r] + [CS].
4). ADC M. (Add memory with carry to accumulator) [A] <-- [A] + [[H-L]] [CS].
5) .ADI data (Add immediate data to accumulator) [A] <-- [A] + data.
6) .ACI data (Add with carry immediate data to accumulator). [A] <-- [A] + data + [CS].
7).DAD rp. (Add register paid to H-L pair). [H-L] <-- [H-L] + [rp].

8).SUB r. (Subtract register from accumulator). [A] <-- [A] – [r].
9).SUB M. (Subtract memory from accumulator). [A] <-- [A] – [[H-L]].
10).SBB r. (Subtract register from accumulator with borrow). [A] <-- [A] – [r] – [CS].
11).SBB M. (Subtract memory from accumulator with borrow). [A] <-- [A] – [[H-L]] – [CS].
12).SUI data. (Subtract immediate data from accumulator) [A] <-- [A] – data.
13).SBI data. (Subtract immediate data from accumulator with borrow). [A] <-- [A] – data – [CS].
14).INR r (Increment register content) [r] <-- [r] +1.
15). INR M. (Increment memory content) [[H-L]] <-- [[H-L]] + 1.
16).DCR r. (Decrement register content). [r] <-- [r] – 1.
17).DCR M. (Decrement memory content) [[H-L]] <-- [[H-L]] – 1.
18).INX rp. (Increment register pair) [rp] <-- [rp] – 1.
19).DCX rp (Decrement register pair) [rp] <-- [rp] -1.
20).DAA (Decimal adjust accumulator) .

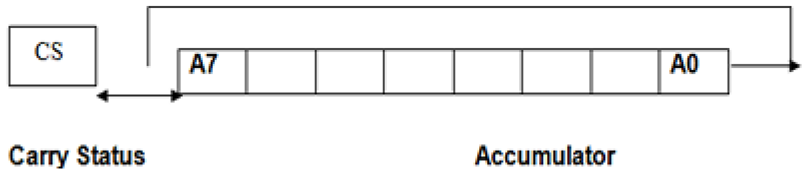
3) Logical instructions:

The Instructions under this group perform logical operation such as AND, OR, compare, rotate etc.
Examples:
1). ANA r. (AND register with accumulator) [A] <-- [A] ^ [r].
2).ANA M. (AND memory with accumulator). [A] <-- [A] ^ [[H-L]].
3).ANI data. (AND immediate data with accumulator) [A] <-- [A] ^ data.
4).ORA r. (OR register with accumulator) [A] <-- [A] v [r].
5).ORA M. (OR memory with accumulator) [A] <-- [A] v [[H-L]]
6).ORI data. (OR immediate data with accumulator) [A] <-- [A] v data.
7).XRA r. (EXCLUSIVE – OR register with accumulator) [A] <-- [A] v [r]
8).XRA M. (EXCLUSIVE-OR memory with accumulator) [A] <-- [A] v [[H-L]]
9).XRI data. (EXCLUSIVE-OR immediate data with accumulator) [A] <-- [A]
10).CMA. (Complement the accumulator) [A] <-- [A]
11).CMC. (Complement the carry status) [CS] <-- [CS]
12).STC. (Set carry status) [CS] <-- 1.
13).CMP r. (Compare register with accumulator) [A] – [r]
14).CMP M. (Compare memory with accumulator) [A] – [[H-L]]
15)CPI data. (Compare immediate data with accumulator) [A] – data.
The 2nd byte of the instruction is data, and it is subtracted from the content of the accumulator. The status flags are set according to the result of subtraction. But the result is discarded. The content of the accumulator remains unchanged.

16).RLC (Rotate accumulator left) [An+1] <-- [An], [A0] <-- [A7],[CS] <-- [A7].



The content of the accumulator is rotated left by one bit. The seventh bit of the accumulator is moved to carry bit as well as to the zero bit of the accumulator. Only CS flag is affected.
17).RRC. (Rotate accumulator right) [A7] <-- [A0], [CS] <-- [A0], [An] <-- [An+1].



The content of the accumulator is rotated right by one bit. The zero bit of the accumulator is moved to the seventh bit as well as to carry bit. Only CS flag is affected.
18). RAL. (Rotate accumulator left through carry) [An+1] <-- [An], [CS] <-- [A7], [A0] <-- [CS].
19).RAR. (Rotate accumulator right through carry) [An] <-- [An+1], [CS] <-- [A0], [A7] <-- [CS].

4) Branching Instructions:

This group includes the instructions for conditional and unconditional jump, subroutine call and return, and restart.
Examples:
1. MP addr (label). (Unconditional jump: jump to the instruction specified by the address). [PC] <-- Label.
2. Conditional Jump addr (label): After the execution of the conditional jump instruction the program jumps to the instruction specified by the address (label) if the specified condition is fulfilled. The program proceeds further in the normal sequence if the specified condition is not fulfilled. If the condition is true and program jumps to the specified label, the execution of a conditional jump takes 3 machine cycles: 10 states. If condition is not true, only 2 machine cycles; 7 states are required for the execution of the instruction.
1. **JZ** addr (label). (Jump if the result is zero)
2. **JNZ** addr (label) (Jump if the result is not zero)
3. **JC** addr (label). (Jump if there is a carry)
4. **JNC** addr (label). (Jump if there is no carry)
5. **JP** addr (label). (Jump if the result is plus)
6. **JM** addr (label). (Jump if the result is minus)
7. **JPE** addr (label) (Jump if even parity)
8. **JPO** addr (label) (Jump if odd parity)
3. CALL addr (label) (Unconditional CALL: call the subroutine identified by the operand)

CALL instruction is used to call a subroutine. Before the control is transferred to the subroutine, the address of the next instruction of the main program is saved in the stack. The content of the stack pointer is decremented by two to indicate the new stack top. Then the program jumps to subroutine starting at address specified by the label.
1. RET (Return from subroutine).
2. RST n (Restart) Restart is a one-word CALL instruction. The content of the program counter is saved in the stack. The program jumps to the instruction starting at restart location.

9 Instruction Set Contd. With sample prog.

5). Stack,I/O and Machine control instructions:
1. N port-address. (Input to accumulator from I/O port) [A] <-- [Port]
2. OUT port-address (Output from accumulator to I/O port) [Port] <-- [A]
3. PUSH rp (Push the content of register pair to stack)
4. PUSH PSW (PUSH Processor Status Word)
5. POP rp (Pop the content of register pair, which was saved, from the stack)
6. POP PSW (Pop Processor Status Word)
7. HLT (Halt)
8. XTHL (Exchange stack-top with H-L)
9. SPHL (Move the contents of H-L pair to stack pointer)
10. EI (Enable Interrupts)
11. DI (Disable Interrupts)
12. SIM (Set Interrupt Masks)
13. RIM (Read Interrupt Masks)
14. NOP (No Operation).

Sample programs of instruction sets:
1).program for adding two 8 bit numbers.
```
 // Load accumulator from memory address
 // Move the content from accumulator to B
    //  Load accumulator from memory address 1001H
    // Add the content of B to accumulator and store the result in accumulator
    // Store the content of accumulator in memory location 1002H
    // Halt
```
Another assembly language instruction sequence to solve the same problem.
```
          // Load lower order byte of address to register
        // Load higher order byte of address to register
        // Move the contents of memory location addressed by pair to
        //  Load the accumulator from memory location
          // Add the content of to accumulator
        // Store the content of accumulator to memory location
```

```
            // Halt.
```

1).program for Multiplying two 8 bit numbers.

```
          MVI A,00  // Load immediate data into accumulator.
          MVI B,02//  Load immediate data into register B.
          MVI C,04// Load immediate data into register C.
LOOP:           ADD B  // Add the content of to accumulator.
          DCR C // Decrement the content of register C by 1.
JNZ  LOOP
          STA 1051H// Store the content of accumulator to memory location 1051H
          HLT    // Halt.
```

Future Technologies

With decades of innovative potential ahead of them, conventional microelectronic designs will dominate much of the 21st century. That trend does not discourage many laboratories from exploring a variety of novel technologies that might be useful in designing new generations of computers and microelectronic devices. In some cases, these approaches would allow chip designs to reach a level of miniaturization unattainable through anything like conventional lithography techniques. Among the ideas being investigated are:

1. **Quantum dots and other single-electron devices.**

Quantum dots are molecular arrays that allow researchers to trap individual electrons and monitor their movements. These devices can in theory be used as binary registers in which the presence or absence of a single electron is used to represent the 0 or 1 of a data bit. In a variation on this scheme, laser light shining on atoms could switch them between their electronic ground state and an excited state, in effect flipping the bit value. One complication of making the transistors and wires extremely small is that quantum-mechanical effects begin to disrupt their function. The logic components hold their 0 or 1 values less reliably because the locations of single electrons become hard to specify. Yet this property could be exploited: Seth Lloyd of the Massachusetts Institute of Technology and other researchers are studying the possibility of developing quantum computing techniques, which would capitalize on the nonclassical behavior of the devices.

2. **Molecular computing.**

Instead of making components out of silicon, some investigators are trying to develop data storage systems using biological molecules. Robert L. Birge of Syracuse University, for example, is examining the computational potential of molecules related to bacteriorhodopsin, a pigment that alters its configuration in response to light. One advantage of such a molecule is that it could be used in an optical computer, in which streams of photons would take the place of electrons. Another is that many of these molecules might be synthesized by microorganisms, rather than fabricated in a factory. According to some estimates, photonically activated biomolecules could be linked into a three-dimensional memory system that would have a capacity 300 times greater than today's CD-ROMs.

3. **Nanomechanical logic gates**.

In these systems, tiny beams or filaments only one atom wide might be physically moved, like Tinkertoys, to carry out logical operations [see "Self-Assembling Materials," by George M. Whitesides, page 146]

4. **Reversible logic gates.**

As the component density on chips rises, dissipating the heat generated by computations becomes more difficult. Researchers at Xerox PARC, the IBM Thomas J. Watson Research Center and elsewhere are therefore checking into the possibility of returning capacitors to their original state at the end of a calculation. Because reversible logic gates would in effect recapture some of the energy expended, they would generate less waste heat.

VI. CONCLUSION

As a microprocessor generation becomes mature and future hardware technologies are better defined, the next generation starts to become visible. We are at that stage now. The last time we were in a similar position was almost 20 years ago—when superscalar processors began to be discussed. In this paper we have surveyed the uniprocessor alternatives, such as advanced superscalar, superspeculative, multiscalar, trace, and datascalar processor, which are all examples of new microarchitectures suitable for the next generation . Developing superspeculative techniques prepares the way for exceeding the classical dataflow limit imposed by data dependences. The multiscalar processor proposed the basics for a functional distribution of processing elements working simultaneously on tasks generated from sequential machine programs. The trace processor is similar to the multiscalar processor except for its use of hardware-generated dynamic traces rather than compiler-generated static tasks. The datascalar approach reduces interprocessor data traffic in favour of broadcasting. A real uniprocessor chip of the future is likely to combine J. Sˇ 188 ilc et al. / Microprocessors and Microsystems 24 (2000) 175–190 Table 1 Summary of new research directions in microprocessors Research direction Key idea Advanced superscalar processor Wide-issue superscalar processing core with speculative execution Superspeculative processor Wide-issue superscalar processing core with aggressive data and control speculation Multiscalar processor Several processing cores speculatively execute different statically generated program segments Trace processor Several processing cores speculatively execute different dynamically generated trace segments Datascalar processor Several processing cores redundantly execute the same sequential program with different data sets some of these execution modes within a single microarchitecture. For instance, an advanced superscalar processor makes use of superspeculative techniques and a trace cache. Recent studies have shown that many instructions perform the same computation and, hence, produce the same result over and over again, i.e. there is significant result redundancy in programs. Two hardware techniques have been proposed to exploit this redundancy value prediction, and instruction reuse. Their impact on superscalar, multiscalar, or trace processors, as well as how to combine value prediction techniques in hybrid value predictors has recently been described in Refs. . All the microarchitecture techniques described increase the memory bandwidth requirements compared to today's superscalar microprocessors. Therefore, all these microarchitecture techniques may be combined in future with the processor-in-memory or intelligent RAM approaches that combine processing elements of various complexity with DRAM memory on the same chip to solve the memory latency bottleneck. All the research directions described in this paper retain result serialization—the serial instruction flow as seen by the programmer and forced by the von Neumann architecture. However, the microarchitectures strive to press as much fine-grained or even coarse-grained parallelism from the sequential program flow as can be achieved by the hardware. Unfortunately, a large portion of the exploited parallelism is speculative parallelism, which in the case of incorrect speculation, leads to an expensive rollback mechanism and to a waste of instruction slots. Therefore, the result serialization of the von Neumann architecture poses a severe bottleneck. Current superscalar microprocessors are able to issue up to six multiple instructions each clock cycle from a conventional linear instruction stream. In this paper we have shown that VLSI technology together with a higher degree of speculation and functional partitioning of the processor will allow future complex uniprocessors with an issue bandwidth of 8–32 instructions per clock cycle. However, instruction-level parallelism found in a conventional instruction stream is limited. In general, integer-dominated programs feature a rather low instruction-level parallelism, while a high instruction-level parallelism can be extracted from floating-point programs. The solution is the additional utilisation of more coarsegrained parallelism. The main approaches are the chip multiprocessor (CMP) and the simultaneous multithreaded processor (SMP). A CMP places a small number of distinct processors (4–16) on a single chip and runs parallel programs and/or multiple independent tasks on these processors. A SMT processor [28,35,36] combines wideissue superscalar processor with multithreading and exhibit high performance increases over single-threaded (superscalar) processors [22]. These two alternatives optimize the throughput of a multiprogramming workload while each thread or process retains the result serialisation of the von Neumann architecture. There are also research directions in highly parallel chip architectures that deviate from the von Neumann architecture model. Two such directions are focused on the processor-in-memory approach (also called intelligent RAM) and on reconfigurable processors. In short, IRAM integrates processor and memory on the same chip to increase memory bandwidth, while reconfigurable processors allow the hardware to adapt dynamically at runtime to the needs of an application.

REFERENCES

[1.] R. Crisp, Direct rambus technology: the new main memory standard, IEEE Micro 17 (1997) 18–28.
[2.] C. Dulong, The IA-64 architecture at work, Computer 31 (1998) 24–31.
[3.] M. Evers, P.-Y. Chang, Y.N. Patt, Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches, in: Proceedings of the ISCA 23, Philadelphia, PA, 1996, pp. 3–11.
[4.] M. Franklin, The multiscalar architecture, Computer Science Technical Report No. 1196, University of Wisconsin-Madison, WI, 1993. [10] F. Gabbay, A. Mendelson, The effect of instruction fetch bandwidth on value prediction, in: Proceedings of the ISCA 25, Barcelona, Spain, 1998, pp. 272–281.
[5.] P. Gillingham, B. Vogley, High-performance, open-standard memory, IEEE Micro 17 (1997) 29–39.
[6.] G. Grohoski, Reining in complexity, Computer 31 (1998) 41–42.
[7.] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.9750&rep=rep1&type=pdf
[8.] http://pdxscholar.library.pdx.edu/cgi/viewcontent.cgi?article=1037&context=etm_fac
[9.] http://www.cs.virginia.edu/~robins/Microprocessors_in_2020.pdf