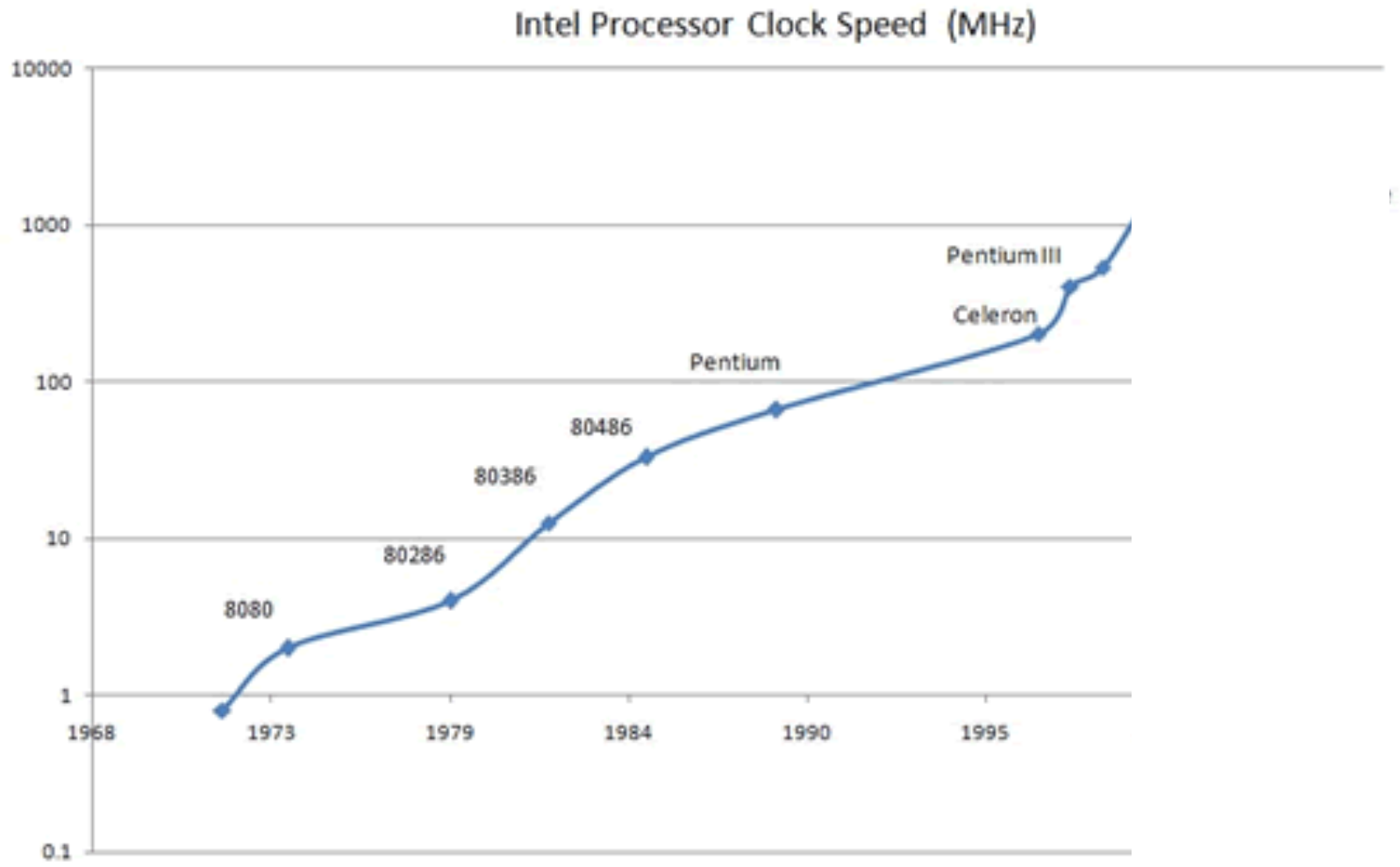
SMPL

A Simple Parallel Language

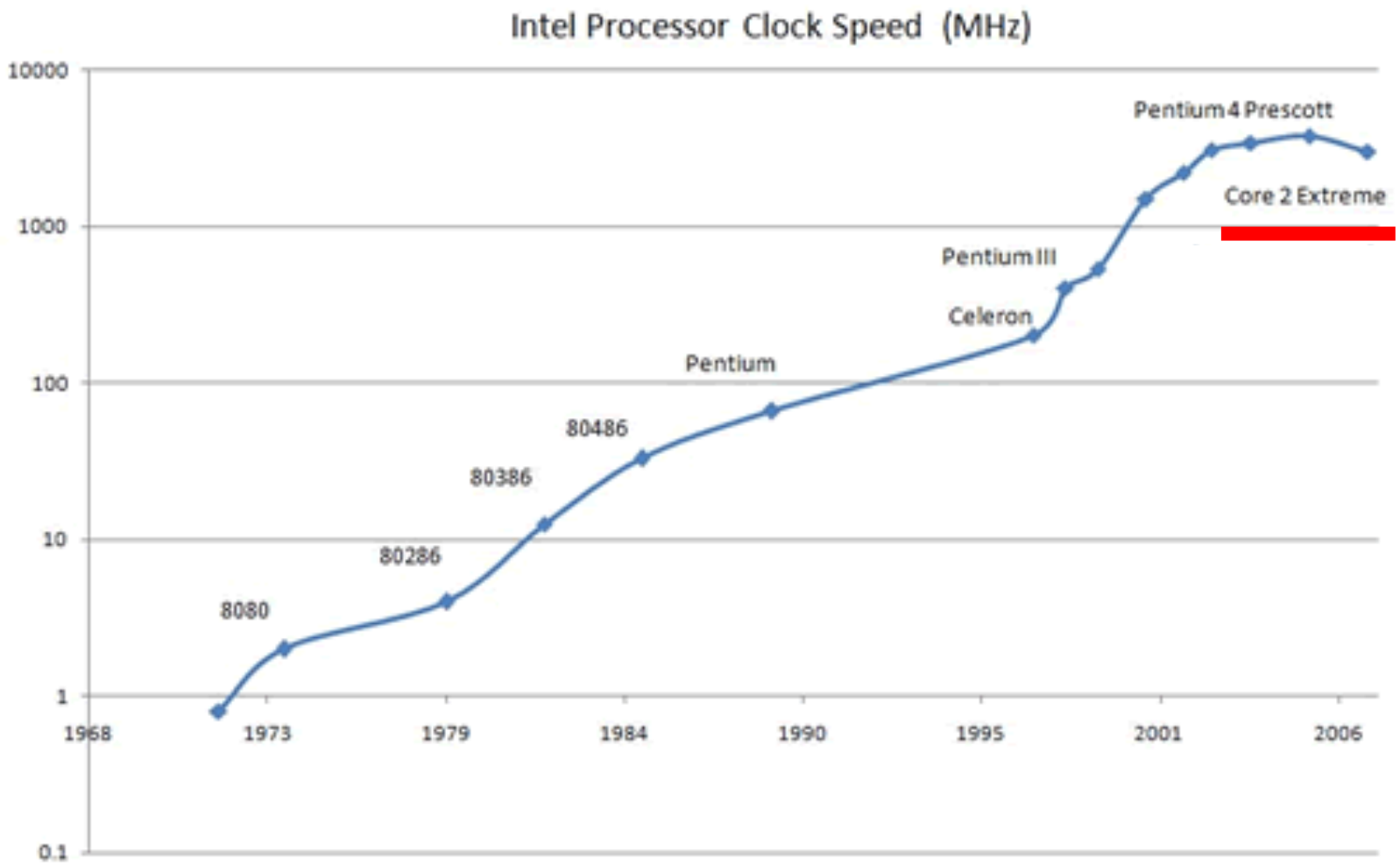
Ajay S S Reddy Challa
Andrei Papancea
Devashi Tandon

Computing over the Years[1]



1 - http://smoothspan.wordpress.com/2007/09/06/a-picture-of-the-multicore-crisis/

Computing over the Years



Gordon E. Moore

- 1965:

  published a paper concerning the future of processor chips

- Moore's Law:

  "the number of transistors on integrated circuits doubles approximately every two years"

The Free Lunch

- Faster processors means faster programs
- No additional effort

SMPL: It’s simple!

- C syntax
- introduces 4 new keywords that allow parallelism
  - spawn
  - barrier
  - lock
  - pfor
- the 4 parallel constructs use Posix threads

SPAWN

- The `spawn` statement creates a thread for the given statement. Its syntax looks as follows:

```
spawn function_call;
```

BARRIER

- The `barrier` statement prevents execution of code following it until all the threads spawned prior to it finish executing. Its syntax looks as follows:

```
barrier;
```

LOCK

- The `lock` statement prevents other threads from accessing or modifying the contents of the statement that it precedes until the latter's computation finishes. Its syntax looks as follows:

```
lock statement
```

PFOR

- The `pfor` statement defines a for loop that splits up the work in its body into multiple threads. Its syntax has the following format:

```
pfor(k; counter; init; limit)
    statement
```

A more exciting “Hello world!”

- This is a quick example on how to use **spawn** and **barrier**:

```
 1 say(string str){
 2   printf(“%s\n”,str);
 3 }
 4
 5 int main(){
 6   spawn say(“Hello”);
 7   spawn say(“world”);
 8   spawn say(“user!”);
 9   barrier;
10   printf(“Done!\n”);
11 }
```

Well, hey there multicore!

- This is a quick example on how to use **pfor**:

```
 1 int sum = 0;
 2
 3 int main(){
 4   int i;
 5   int n = 1000000;
 6
 7   pfor(8; i; 1; n){
 8     sum = sum+i;
 9   }
10
11   printf("The sum of the first 1M
12           integers is %d.\n",sum);
13 }
```

Lock the vault!

- This is a quick example on how to use **lock**:

```
1  float balance = 2000.00;
2
3  withdraw(float val){
4    lock {
5      if(balance-val >= 0)
6        balance = balance-val;
7    }
8  }
9
10 int main(){
11   int i;
12   int n = 1000000;
13   for(i=1; i<100; i++){
14     float amount = i*10;
15     spawn withdraw(amount);
16   }
17   printf("The remaining balance is %f.\n",balance);
18 }
```

Implementation

- semantic checker
  - automatic type casting
  - code validation
  - optimization (reference count)
- code generation
  - C code generation
  - optimization (remove dead code)
- testing
  - check syntax
  - check semantics (manually)
  - check program execution

Lessons learned

- Ajay
  - meeting regularly is crucial
  - keeping SMPL simple helped in the development process
- Andrei
  - start early
  - OCaml is extremely annoying at first, yet extremely powerful
  - do not attempt to implement 10,000 language features
- Devashi
  - coming up with a new language was fruitful and tricky
  - writing a compiler in a completely new language was challenging
  - I learned how to effectively work in a team

DEMO