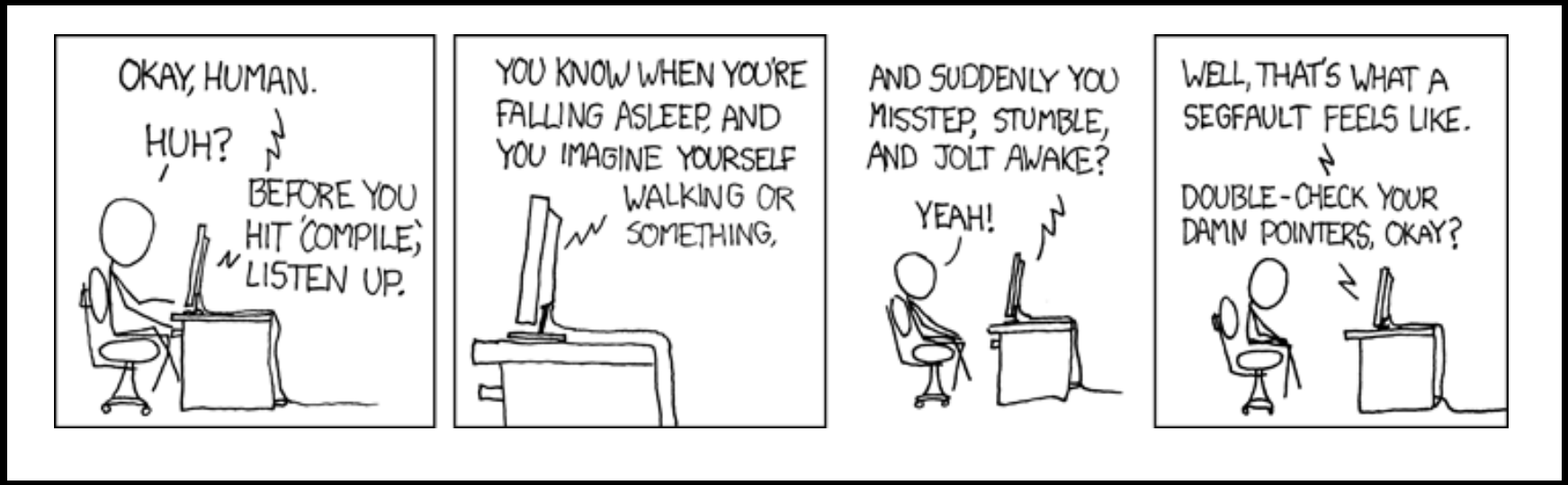
EECS 370 Discussion



xkcd.com

EECS 370 Discussion

Topics Today:

- Control Hazards
- Branch Prediction
- Project 3
- stackoverflow Example

EECS 370 Discussion

Control Hazards

Key Concept

Which LC-2K instruction(s) can cause a Control Hazard?

In which stage are branches resolved?

EECS 370 Discussion

Control Hazards

Key Concept

Which LC-2K instruction(s) can cause a Control Hazard?

BEQ & JALR

In which stage are branches resolved?

MEM

EECS 370 Discussion

Control Hazards

Problem:

If we don't know what the next PC should be what do we do?

Options:

- No Branches
- Avoid
- Detect-and-stall
- Speculate-and-squash

EECS 370 Discussion

Control Hazards

1) No Branches

Is this a feasible solution?

How could we eliminate this if statement?

```
if (r0 == r1) {
     r2 = r3;
} else {
     r2++;
}
```

EECS 370 Discussion

Control Hazards

1) No Branches

Is this a feasible solution?

How could we eliminate this if statement?

```
if (r0 == r1) {
     r2 = r3;
} else {
     r2++;
}
```

Conditional Assembly!

```
cmp   r0, r1
moveq r2, r3
addne r2, r2, #1
```

EECS 370 Discussion

Control Hazards

2) Avoid

```
      ADD  1 1 1
      NAND 2 2 2
      BEQ  3 0 jump
      ADD  4 4 4
      NAND 5 5 5
jump  ADD  6 6 6
      NAND 7 7 7
```

EECS 370 Discussion

Control Hazards

3) Detect-and-stall

Any better than avoid?

EECS 370 Discussion

Control Hazards

4) Speculate-and-squash

Guess!

What do you have to do if you're correct?

What do you have to do if you're wrong?

EECS 370 Discussion

Control Hazards

4) Speculate-and-squash

Guess!

What do you have to do if you're correct?

Nothing at all

What do you have to do if you're wrong?

Turn IF, ID, & EX into Noops

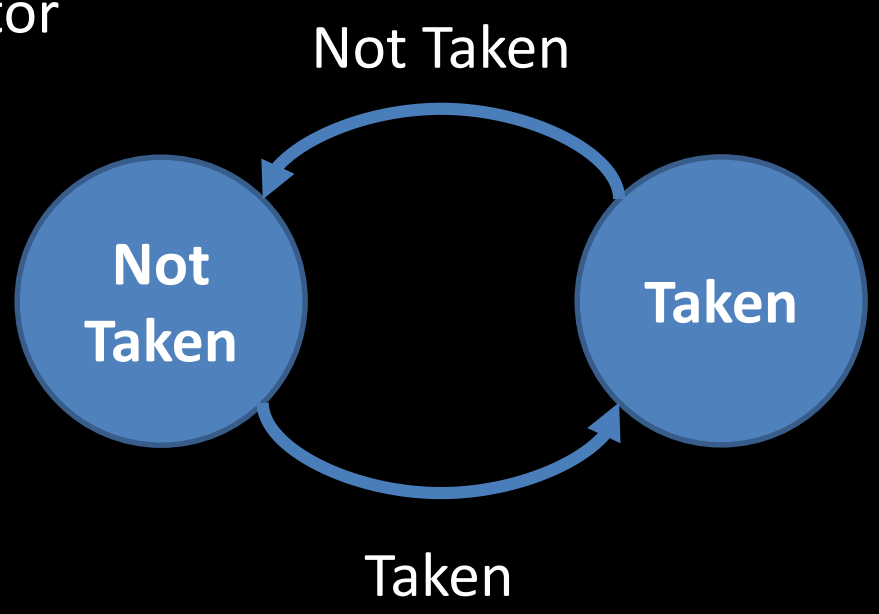
EECS 370 Discussion

Branch Prediction

Based on the PC

What did this branch do last time?

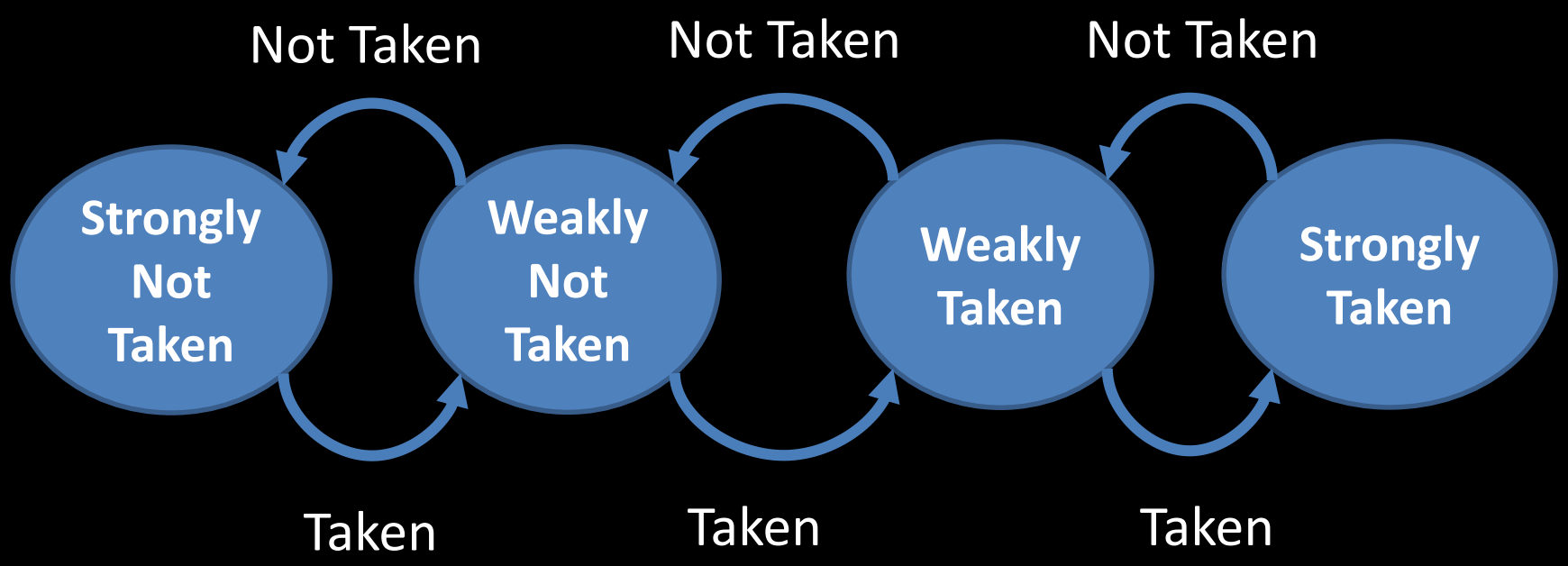
1-bit predictor



EECS 370 Discussion

Branch Prediction

2-bit predictor



EECS 370 Discussion

Branch Prediction

What does the PC become if we predict Not Taken?

What does the PC become if we predict Taken?

EECS 370 Discussion

Branch Prediction

What does the PC become if we predict Not Taken?

PC+1

What does the PC become if we predict Taken?

???

EECS 370 Discussion

Branch Prediction

Branch Target Buffer

Maps PC values to Addresses

PC	Address
0x1234	0x00001000
0x4523	0x00004000
0xA342	0x00000004
0xFF76	0x0000A342

EECS 370 Discussion

Branch Prediction

PC

PC	Address
0x1234	
0x4523	
0xA342	
0xFF76	

PC	Address
0x1234	0x00001000
0x4523	0x00004000
0xA342	0x00000004
0xFF76	0x0000A342

Do we branch?

Where?

EECS 370 Discussion

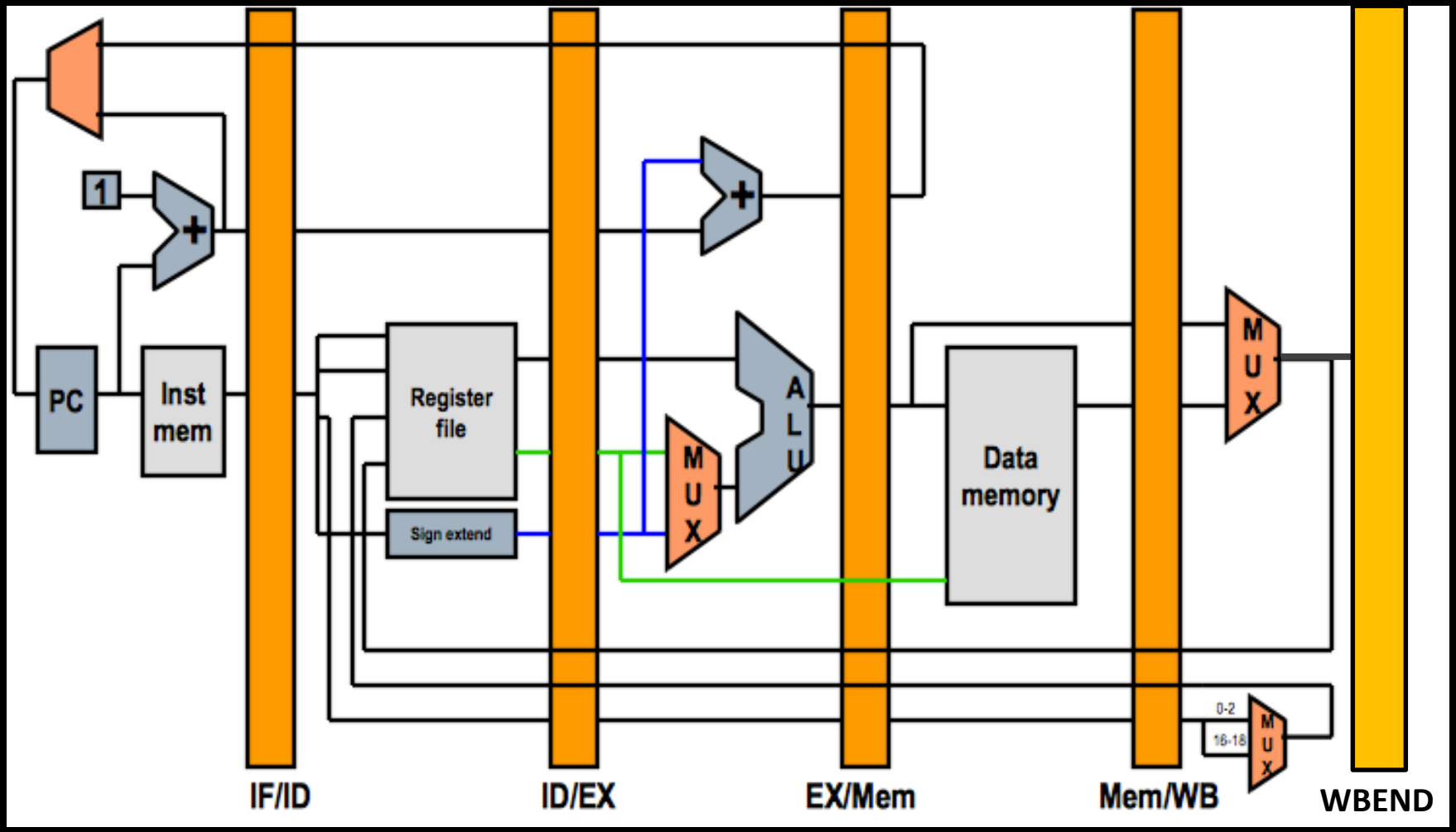
Project 3

Make your own pipelined processor

- Need to use code for the file input from Project 1
- Write non-hazard code first
- Design good test code!!

EECS 370 Discussion

WBEND Register and Forwarding



EECS 370 Discussion

stackoverflow example

Real-world example. Why this stuff matters.

http://stackoverflow.com/questions/11227809/why-is-processing-a-sorted-array-faster-than-an-unsorted-array

```java
public class Main
{
    public static void main(String[] args)
    {
        // Generate data
        int arraySize = 32768;
        int data[] = new int[arraySize];

        Random rnd = new Random(0);
        for (int c = 0; c < arraySize; ++c)
            data[c] = rnd.nextInt() % 256;

        // !!! With this, the next loop runs faster
        Arrays.sort(data);

        // Test
        long start = System.nanoTime();
        long sum = 0;

        for (int i = 0; i < 100000; ++i)
        {
            // Primary loop
            for (int c = 0; c < arraySize; ++c)
            {
                if (data[c] >= 128)
                    sum += data[c];
            }
        }

        System.out.println((System.nanoTime() - start) / 1000000000.0);
        System.out.println("sum = " + sum);
    }
}
```

```java
public class Main
{
    public static void main(String[] args)
    {
        // Generate data
        int arraySize = 32768;
        int data[] = new int[arraySize];

        Random rnd = new Random(0);
        for (int c = 0; c < arraySize; ++c)
            data[c] = rnd.nextInt() % 256;

        // !!! With this, the next loop runs faster
        Arrays.sort(data);

        // Test
        long start = System.nanoTime();
        long sum = 0;

        for (int i = 0; i < 100000; ++i)
        {
            // Primary loop
            for (int c = 0; c < arraySize; ++c)
            {
                if (data[c] >= 128)
                    sum += data[c];
            }
        }

        System.out.println((System.nanoTime() - start) / 1000000000.0);
        System.out.println("sum = " + sum);
    }
}
```

With this line: 6.54 seconds
Without: 13.84 seconds

```
int main()
{
    // Generate data
    const unsigned arraySize = 32768;
    int data[arraySize];

    for (unsigned c = 0; c < arraySize; ++c)
        data[c] = std::rand() % 256;

    // !!! With this, the next loop runs faster
    std::sort(data, data + arraySize);

    // Test
    clock_t start = clock();
    long long sum = 0;

    for (unsigned i = 0; i < 100000; ++i)
    {
        // Primary loop
        for (unsigned c = 0; c < arraySize; ++c)
        {
            if (data[c] >= 128)
                sum += data[c];
        }
    }

    double elapsedTime = static_cast<double>(clock() - start) / CLOCKS_PER_SEC;

    std::cout << elapsedTime << std::endl;
    std::cout << "sum = " << sum << std::endl;
}
```

With this line: 1.93 seconds
Without: 11.54 seconds

```
int main()
{
    // Generate data
    const unsigned arraySize = 32768;
    int data[arraySize];

    for (unsigned c = 0; c < arraySize; ++c)
        data[c] = std::rand() % 256;

    // !!! With this, the next loop runs faster
    std::sort(data, data + arraySize);

    // Test
    clock_t start = clock();
    long long sum = 0;

    for (unsigned i = 0; i < 100000; ++i)
    {
        // Primary loop
        for (unsigned c = 0; c < arraySize; ++c)
        {
            if (data[c] >= 128)
                sum += data[c];
        }
    }

    double elapsedTime = static_cast<double>(clock() - start) / CLOCKS_PER_SEC;

    std::cout << elapsedTime << std::endl;
    std::cout << "sum = " << sum << std::endl;
}
```

Why is this happening? What is causing the time differences?

```
int main()
{
    // Generate data
    const unsigned arraySize = 32768;
    int data[arraySize];

    for (unsigned c = 0; c < arraySize; ++c)
        data[c] = std::rand() % 256;

    // !!! With this, the next loop runs faster
    std::sort(data, data + arraySize);

    // Test
    clock_t start = clock();
    long long sum = 0;

    for (unsigned i = 0; i < 100000; ++i)
    {
        // Primary loop
        for (unsigned c = 0; c < arraySize; ++c)
        {
            if (data[c] >= 128)
                sum += data[c];
        }
    }

    double elapsedTime = static_cast<double>(clock() - start) / CLOCKS_PER_SEC;

    std::cout << elapsedTime << std::endl;
    std::cout << "sum = " << sum << std::endl;
}
```

Branch Prediction!