Ford as a Software Company


2 Cycle development
• **Stalling** an instruction = hold it back in the pipeline, including all instructions behind it
• When stalling, need to inject a **bubble** in place of the stalled instruction

• bomb code $\rightarrow$ next lab period

• memory mountain: see pp. 621-624
  
  “It is not entirely clear why these dips occur. The only way to be sure is to perform a detailed cache simulation…..”
Summary of Matrix Multiplication

**ijk (& jik):**
- 2 loads, 0 stores
- misses/iter = 1.25

```plaintext
for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
        sum = 0.0;
        for (k=0; k<n; k++)
            sum += a[i][k] * b[k][j];
        c[i][j] = sum;
    }
}
```

**kij (& ikj):**
- 2 loads, 1 store
- misses/iter = 0.5

```plaintext
for (k=0; k<n; k++) {
    for (i=0; i<n; i++) {
        r = a[i][k];
        for (j=0; j<n; j++)
            c[i][j] += r * b[k][j];
    }
}
```

**jki (& kji):**
- 2 loads, 1 store
- misses/iter = 2.0

```plaintext
for (j=0; j<n; j++) {
    for (k=0; k<n; k++) {
        r = b[k][j];
        for (i=0; i<n; i++)
            c[i][j] += a[i][k] * r;
    }
}
```
## Practice: Which is better and why?

```c
#define N 1000

typedef struct {
    int vel[3];
    int acc[3];
} point;

point p[N];

void clear1 (point *p, int n) {
    int i, j;
    for (i=0; i<n; i++){
        for (j=0; j<3; j++)
            p[i].vel[j] = 0;
        for (j=0; j<3; j++)
            p[i].acc[j]=0;
    }
}

void clear3 (point *p, int n) {
    int i, j;
    for (j=0; j<3; j++){
        for (i=0; i<n; i++)
            p[i].vel[j] = 0;
        for (i=0; i<n; i++)
            p[i].acc[j]=0;
    }
}
```
General Cache Organization \((S, E, B)\)

- \(E = 2^e\) lines per set
- \(S = 2^s\) sets
- \(B = 2^b\) bytes per cache block (the data)

Cache size: \(S \times E \times B\) data bytes
Cache

• Direct-mapped cache (E=1):
  – Primary cache memory has $S$ blocks of memory of $B$ size.
  – Secondary memory has $S'$ blocks of memory of $B$ size.
  – $S'$ is a multiple of $S$.
  – Each block will map into a specific cache location based on the block number modulus $S$.
  – If a block exists in a specific location, then it will be evicted.
Example: Direct Mapped Cache (E = 1)

Direct mapped: One line per set
Assume: cache block size 8 bytes

$S = 2^s$ sets

Address of int:

| t bits | 0...01 | 100 |

Find set
Example: Direct Mapped Cache \((E = 1)\)

Direct mapped: One line per set
Assume: cache block size 8 bytes

<table>
<thead>
<tr>
<th>V</th>
<th>Tag</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>

Address of int:

- **t bits**: \(0...01\) 100

valid? + match: assume yes = hit

Block offset

No match: old line is evicted and replaced
2\textsuperscript{nd} (Colorful) Example

Size of this Cache?
Size of this Main Memory?
2\textsuperscript{nd} (Colorful) Example

Main Memory contains:

\[ 2^5 \times 2^8 \times 2^3 = 2^{16} \text{Bytes} = 64\text{KB} \]

Cache Memory contains:

\[ 2^8 \times 2^3 = 2^{11} \text{Bytes} = 2\text{KB} \]
3rd Example

4 sets, one line per block, 2 bytes per block, 4-bit addresses

<table>
<thead>
<tr>
<th>Address (decimal)</th>
<th>Tag bit ((t=1))</th>
<th>Index bits ((s=2))</th>
<th>Offset bits ((b=1))</th>
<th>Block number (decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>01</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>00</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>00</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>01</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>01</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Read at address 0
Read 1
Read 13
Read 8
Read 0