x86-64 Programming IV

CSE 351 Spring 2024

Instructor:

Elba Garza

Teaching Assistants:

Ellis Haker	Maggie Jiang
Adithi Raghavan	Malak Zaki
Aman Mohammed	Naama Amiel
Brenden Page	Nikolas McNamee
Celestine Buendia	Shananda Dokka
Chloe Fong	Stephen Ying
Claire Wang	Will Robertson
Hamsa Shankar	

Me multiplying two numbers in Assembly
RollerCoaster Tycoon
Chris Sawyer, who coded the entirety of Rollercoaster Tycoon by himself in Assembly

Announcements, Reminders

- Lab 1b & HW7 due tonight by 11:59 PM!
- You will need to use GDB to get through Lab 2
  - Useful debugger in this class and beyond!

    Tips: https://courses.cs.washington.edu/courses/cse351/24sp/debug/
  - Also, GDB Demo Video on Ed
  - This week's section will also have some Lab 2 prep, so take advantage!
- Mid-Quarter Evaluation: April 24th, in class
- Midterm: May 6th for 48 hours
  - Week 6's section (02 May) will be for midterm review 😎

Choosing instructions for conditionals

		`cmp a,b`	`test a,b`
`je`	"Equal"	`b == a`	`b&a == 0`
`jne`	"Not equal"	`b != a`	`b&a != 0`
`js`	"Sign" (negative)	`b-a < 0`	`b&a < 0`
`jns`	(non-negative)	`b-a >=0`	`b&a >= 0`
`jg`	"Greater"	`b > a`	`b&a > 0`
`jge`	"Greater or equal"	`b >= a`	`b&a >= 0`
`jl`	"Less"	`b < a`	`b&a < 0`
`jle`	"Less or equal"	`b <= a`	`b&a <= 0`
`ja`	"Above" (unsigned >)	`b` $>_U$ `a`	`b&a > 0U`
`jb`	"Below" (unsigned <)	`b` $<_U$ `a`	`b&a < 0U`

Register	Use(s)
`%rdi`	1st argument (`x`)
`rsi`	2nd argument (`y`)
`%rax`	return value

```
if (x < 3) {
  return 1;
}
return 2;
```

```
  cmpq $3, %rdi
  jge T2
T1: # x < 3:
  movq $1, %rax
  ret
T2: # !(x < 3):
  movq $2, %rax
  ret
```

Choosing instructions for conditionals

https://godbolt.org/z/Tfrv33

		`cmp a,b`	`test a,b`
`je`	"Equal"	`b == a`	`b&a == 0`
`jne`	"Not equal"	`b != a`	`b&a != 0`
`js`	"Sign" (negative)	`b-a < 0`	`b&a < 0`
`jns`	(non-negative)	`b-a >=0`	`b&a >= 0`
`jg`	"Greater"	`b > a`	`b&a > 0`
`jge`	"Greater or equal"	`b >= a`	`b&a >= 0`
`jl`	"Less"	`b < a`	`b&a < 0`
`jle`	"Less or equal"	`b <= a`	`b&a <= 0`
`ja`	"Above" (unsigned >)	$b >_U a$	`b&a > 0U`
`jb`	"Below" (unsigned <)	$b <_U a$	`b&a < 0U`

```
if (x < 3 && x == y) {
  return 1;
} else {
  return 2;
}
```

```
  cmpq $3, %rdi
  setl %al        // < (SF^OF)

  cmpq %rsi, %rdi
  sete %bl        // == 0 (ZF)

  testb %al, %bl
  je T2           // == 0 (ZF)
T1: # x < 3 && x == y:
   movq $1, %rax
   ret
T2: # else
   movq $2, %rax
   ret
```

Reading Review

- Terminology:
  - Label, jump target
  - Program counter
  - Jump table, indirect jump

Labels

```
swap:
    movq  (%rdi), %rax
    movq  (%rsi), %rdx
    movq  %rdx, (%rdi)
    movq  %rax, (%rsi)
    ret
```

```
max:
    movq %rdi, %rax
    cmpq %rsi, %rdi
    jg   done
    movq %rsi, %rax
done:
    ret
```

- A jump changes the program counter (`%rip`)
  - `%rip` tells the CPU the address of the next instruction to execute
- **Labels** give us a way to refer to a specific instruction in our assembly/machine code
  - Associated with the **next** instruction found in the assembly code (ignores whitespace)
  - Each **use** of the label will eventually be replaced with something that indicates the final address of the instruction that it is associated with

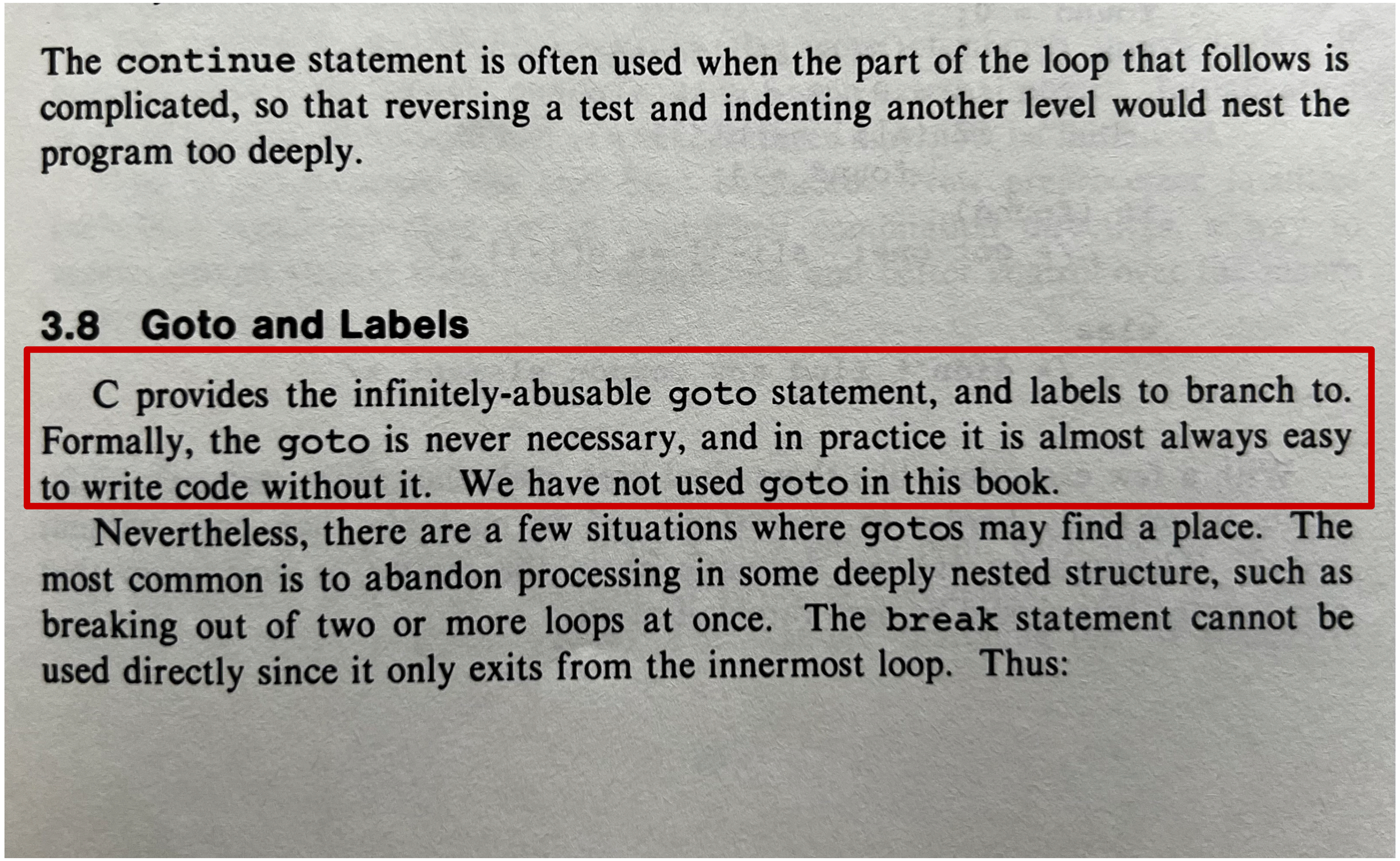
Aside: Labels & Jumps in C (goto)

```
long absdiff(long x, long y)
{
    long result;
    if (x > y)
        result = x-y;
    else
        result = y-x;
    return result;
}
```

```
long absdiff_j(long x, long y)
{
    long result;
    int ntest = (x <= y);
    if (ntest) goto Else;
    result = x-y;
    goto Done;
 Else:
    result = y-x;
 Done:
    return result;
}
```

- C allows **goto** as means of transferring control
  - Closer to assembly programming style
  - Don’t do this!! Bad!!! But if you won’t listen to us, listen to K&R…

Aside: Labels & Jumps in C (goto)


The continue statement is often used when the part of the loop that follows is complicated, so that reversing a test and indenting another level would nest the program too deeply.

3.8 Goto and Labels

C provides the infinitely-abusable goto statement, and labels to branch to. Formally, the goto is never necessary, and in practice it is almost always easy to write code without it. We have not used goto in this book.

Nevertheless, there are a few situations where gotos may find a place. The most common is to abandon processing in some deeply nested structure, such as breaking out of two or more loops at once. The break statement cannot be used directly since it only exits from the innermost loop. Thus:

x86 Control Flow

- Condition codes
- Conditional and unconditional branches
- **Loops**
- Switches

Compiling Loops

C/Java code:

```
while ( sum != 0 ) {
    <loop body>
}
```

Assembly code:

```
loopTop:    testq %rax, %rax   # test variable sum
            je    loopDone     # test inverse of condition
            <loop body code>
            jmp   loopTop      # unconditional jump back!
loopDone:
```

- Other loops compiled similarly
  - Will show variations and complications in coming slides, but may skip a few examples in the interest of time
- Most important to consider:
  - When should conditionals be evaluated? (*while* vs. *do-while*)
  - How much jumping is involved?

Compiling Loops

While Loop:

C:
```
while ( sum != 0 ) {
    <loop body>
}
```

x86-64:
```
loopTop:    testq %rax, %rax
            je    loopDone
            <loop body code>
            jmp   loopTop
loopDone:
```

Do-while Loop:

C:
```
do {
    <loop body>
} while ( sum != 0 )
```

x86-64:
```
loopTop:
            <loop body code>
            testq  %rax, %rax
            jne    loopTop
loopDone:
```

While Loop (ver. 2):

C:
```
while ( sum != 0 ) {
    <loop body>
}
```

x86-64:
```
            testq %rax, %rax
            je    loopDone
loopTop:
            <loop body code>
            testq %rax, %rax
            jne   loopTop
loopDone:
```

For-Loop → While-Loop

For-Loop:

```
for (Init; Test; Update) {
      Body
}
```

While-Loop Version:

```
Init;
while (Test) {
      Body
      Update;
}
```

Caveat: C and Java have `break` and `continue`

- Conversion works fine for `break`
  - Jump to same label as loop exit condition
- But not `continue`: would skip doing the *Update*, which it should do with for-loops
  - Must introduce new label at *Update*!

Practice Question

- The following is assembly code for a for-loop; identify the corresponding parts (Init, Test, Update)

Register	Use(s)
`%eax`	`i`
`%rdi`	`x`
`%esi`	`y`

```
Line
 1          movl    $0, %eax
 2   .L2:   cmpl    %esi, %eax
 3          jge     .L4
 4          movslq  %eax, %rdx
 5          leaq    (%rdi,%rdx,4), %rcx
 6          movl    (%rcx), %edx
 7          addl    $1, %edx
 8          movl    %edx, (%rcx)
 9          addl    $1, %eax
10          jmp     .L2
11   .L4:
```

Init: Line #____ **Test:** Lines #____ **Update:** Line #____

```
for (            ;                ;              ) {…}
```

How did we get here?

- Loops: do the same thing over and over and over again
- Distinction between creating and running programs
  - Values inform which is "better" and which is "worse"
- Modern hardware, modern "house", historical relics
- Decisions were not an accident!
  - Priorities may have been dated, inconsistent, prejudiced

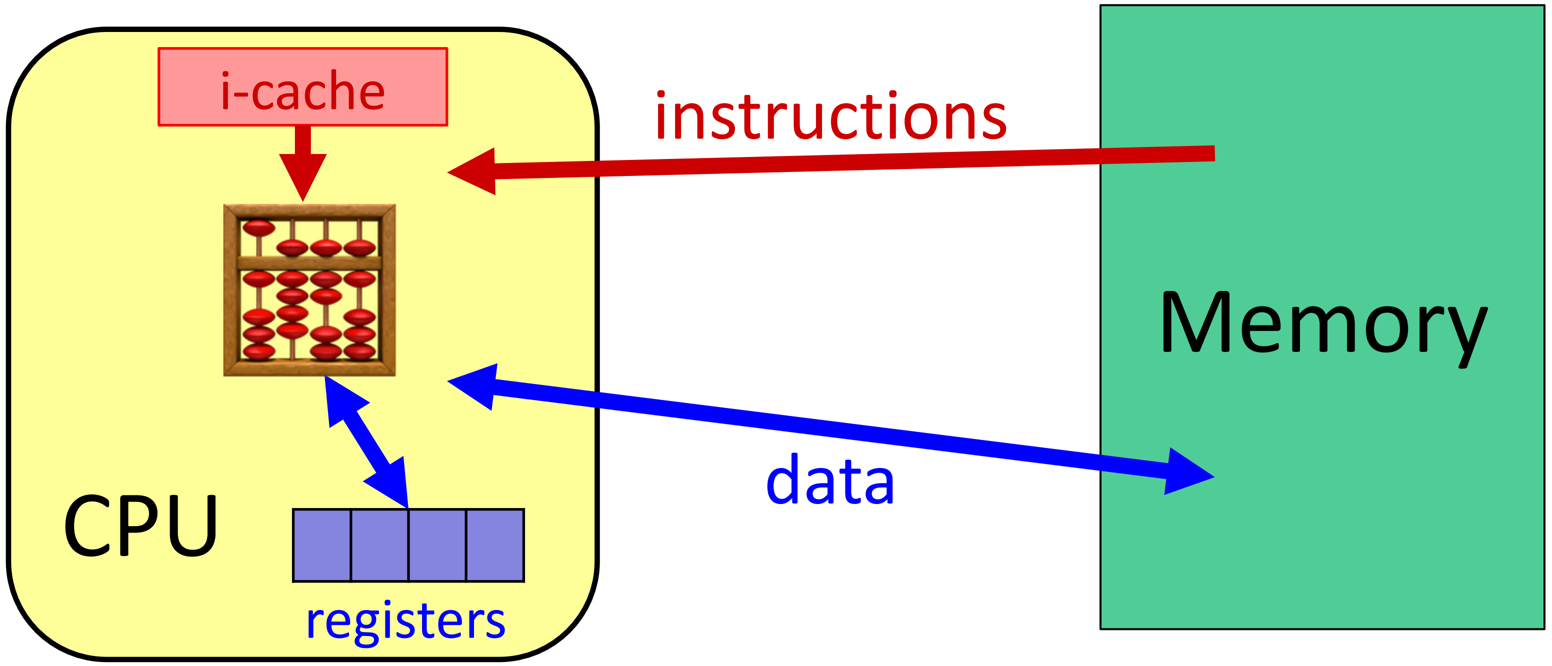
- First: Who acted as the first computers?
- Also: What were the prevailing narratives that informed computing during its modern* incarnation?

* starting in the early 20th century

Prevailing Narratives in Computer Science

- **“Boring, repetitive work” should be automated or augmented for efficiency and profit**
- “Boring, repetitive work” is “robot work”
- Augmentation is highly valued and exclusive

Hardware: 351 View (version 1)



- More CPU details:
  - Instructions are held temporarily in the instruction cache
  - Other data are held temporarily in registers
- Instruction fetching is hardware-controlled
- Data movement is programmer-controlled (assembly)

(Modern) Hardware: Historic View

- **Computer**: one who computes

The women of Bletchley Park, Credit: BBC

- Mostly single wealthy women
- "Boring, repetitive work", doing math quickly

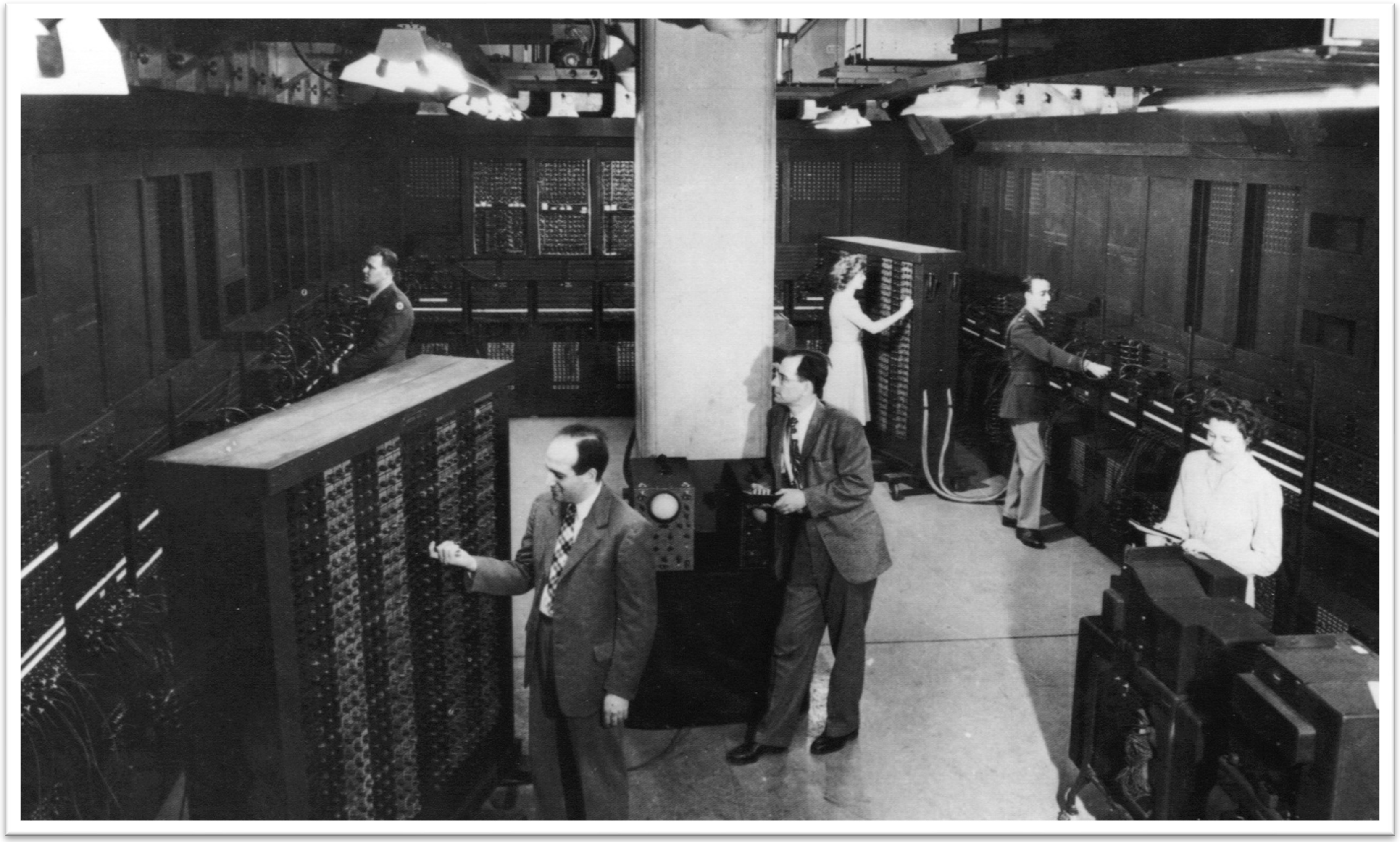
Computing in the US

- **Computer**: one who computes
- Observatory calculations @ Harvard (1870s)

Human Computers at NACA, Credit: NASA

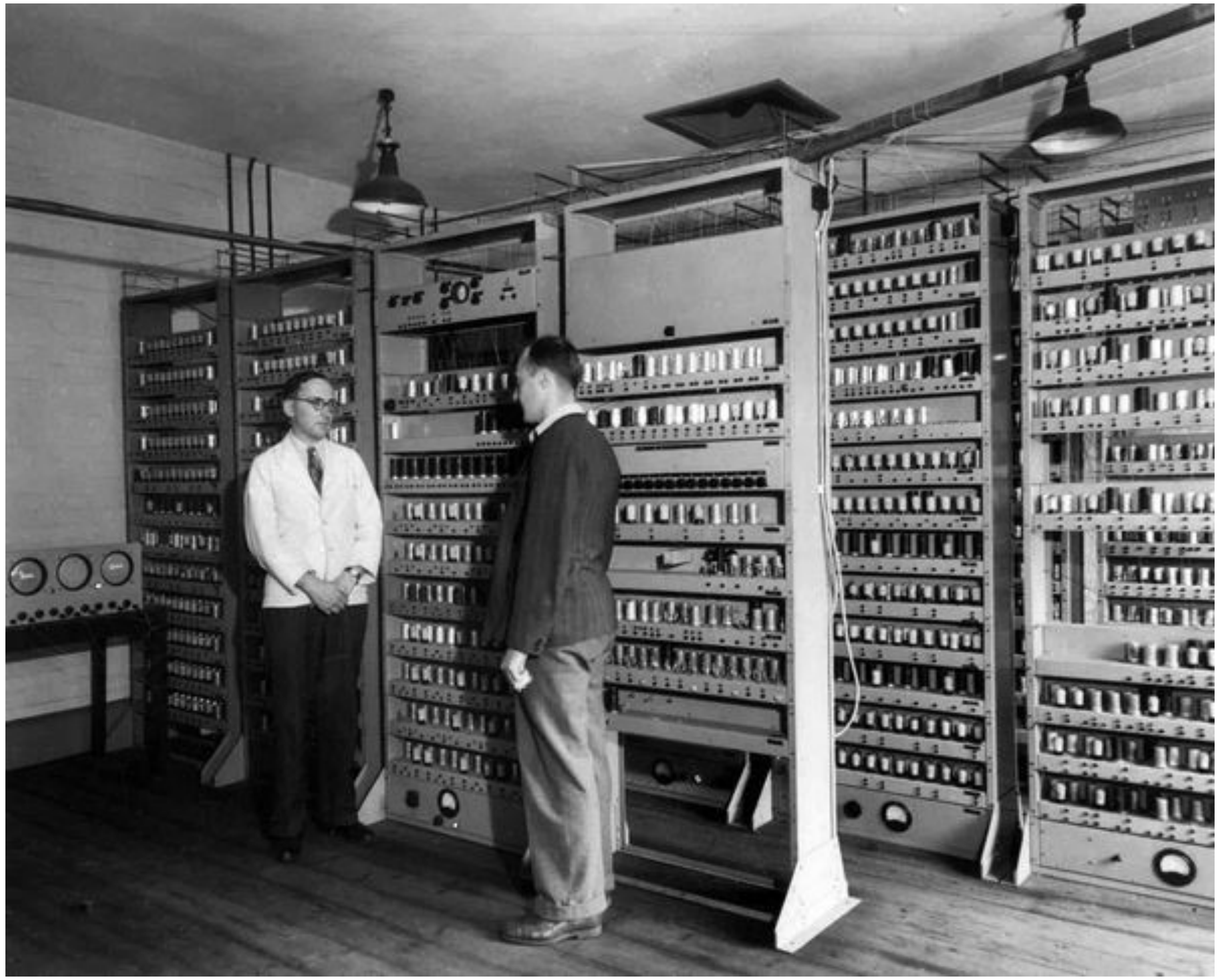
Human Computers at JPL, Credit: JPL

ENIAC (1945): Augmenting & Automating



ENIAC (1945): Augmenting & Automating

EDSAC (1949): Same Thing, Different Continent

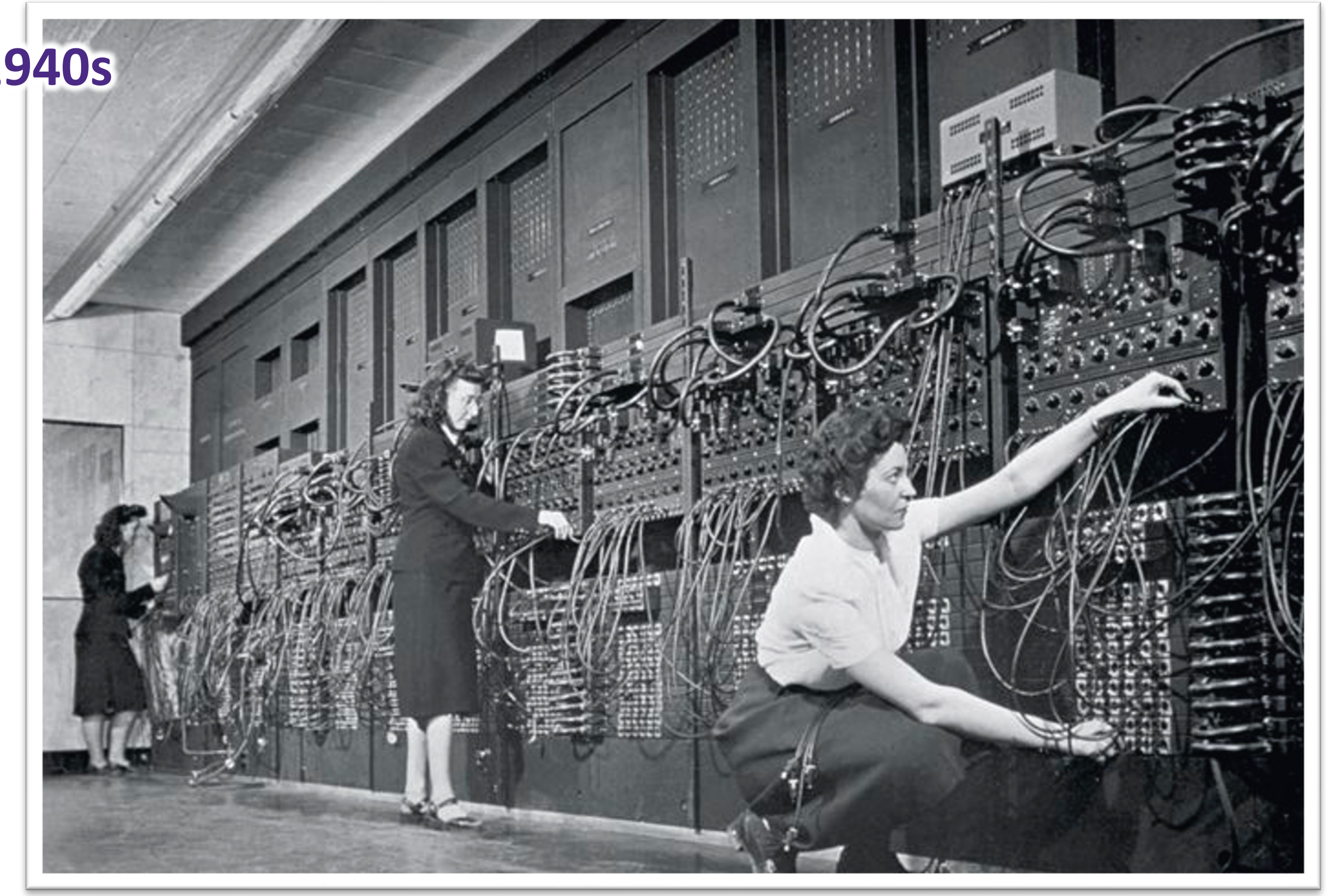


Electronic
Delay
Storage
Automatic
Calculator

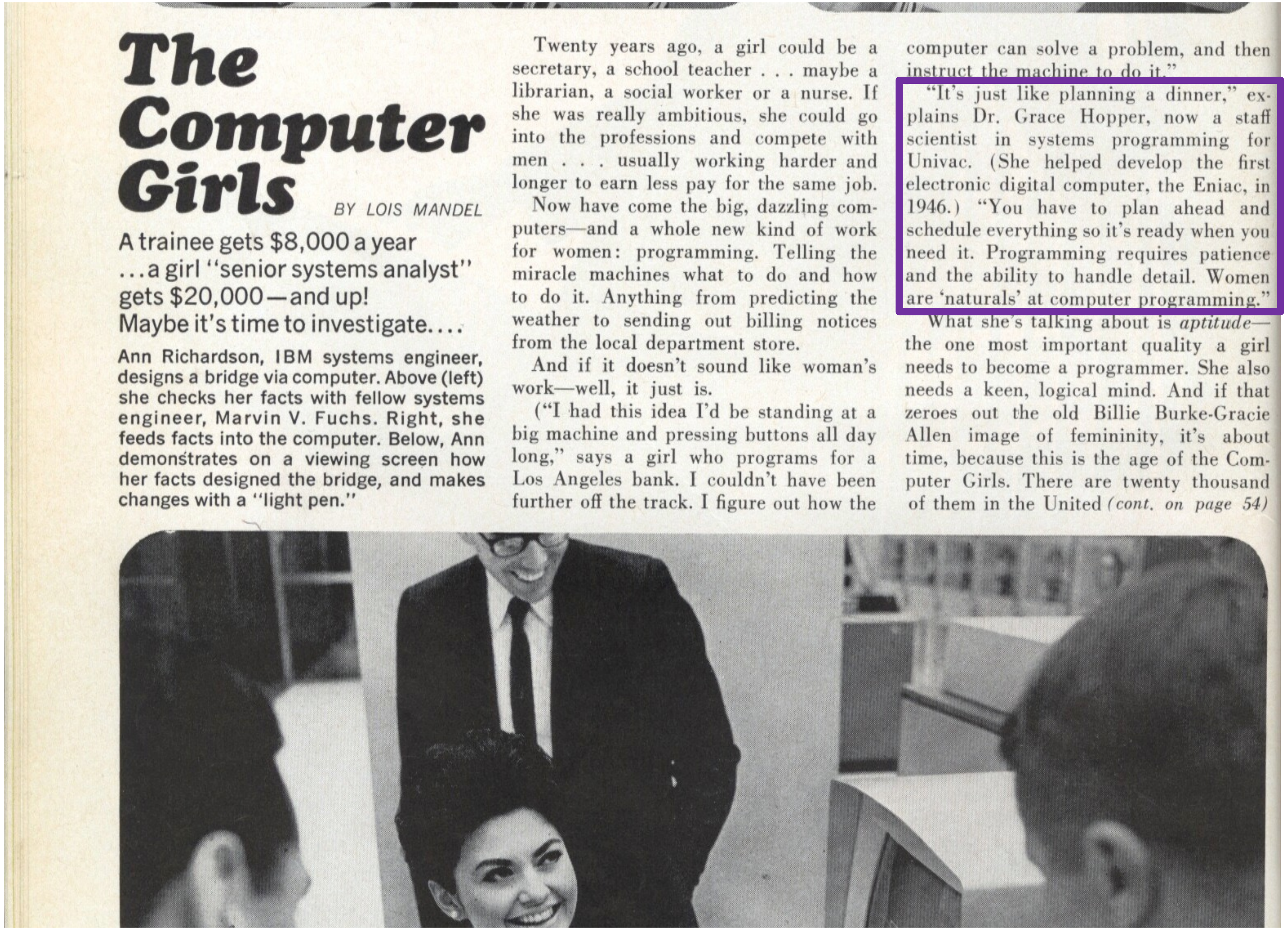
EDSAC (1949): Same Thing, Different Continent

Historical View of Programming

1940s



Jean Jennings (left), Marlyn Wescoff (center), and Ruth Lichterman program ENIAC at the University of Pennsylvania, circa 1946.
Photo: Corbis
http://fortune.com/2014/09/18/walter-isaacson-the-women-of-eniac/



The Computer Girls

BY LOIS MANDEL

A trainee gets $8,000 a year ...a girl "senior systems analyst" gets $20,000—and up! Maybe it's time to investigate....

Ann Richardson, IBM systems engineer, designs a bridge via computer. Above (left) she checks her facts with fellow systems engineer, Marvin V. Fuchs. Right, she feeds facts into the computer. Below, Ann demonstrates on a viewing screen how her facts designed the bridge, and makes changes with a "light pen."

Twenty years ago, a girl could be a secretary, a school teacher . . . maybe a librarian, a social worker or a nurse. If she was really ambitious, she could go into the professions and compete with men . . . usually working harder and longer to earn less pay for the same job.

Now have come the big, dazzling computers—and a whole new kind of work for women: programming. Telling the miracle machines what to do and how to do it. Anything from predicting the weather to sending out billing notices from the local department store.

And if it doesn't sound like woman's work—well, it just is.

("I had this idea I'd be standing at a big machine and pressing buttons all day long," says a girl who programs for a Los Angeles bank. I couldn't have been further off the track. I figure out how the computer can solve a problem, and then instruct the machine to do it.")

"It's just like planning a dinner," explains Dr. Grace Hopper, now a staff scientist in systems programming for Univac. (She helped develop the first electronic digital computer, the Eniac, in 1946.) "You have to plan ahead and schedule everything so it's ready when you need it. Programming requires patience and the ability to handle detail. Women are 'naturals' at computer programming."

What she's talking about is *aptitude*—the one most important quality a girl needs to become a programmer. She also needs a keen, logical mind. And if that zeroes out the old Billie Burke-Gracie Allen image of femininity, it's about time, because this is the age of the Computer Girls. There are twenty thousand of them in the United *(cont. on page 54)*

Prevailing Narratives in Computer Science

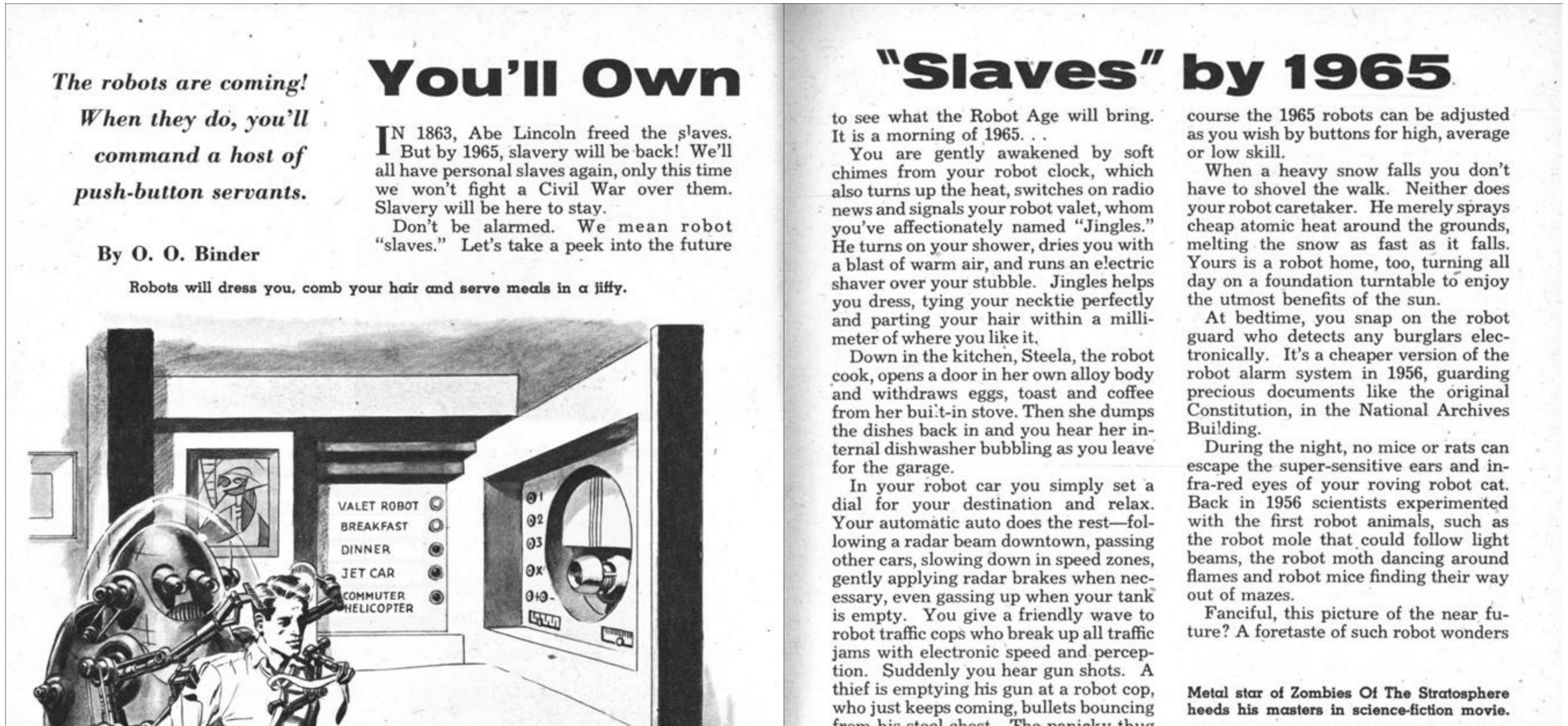
- **“Boring, repetitive work” should be automated or augmented for efficiency and profit**
- “Boring, repetitive work” is “robot work”
- Augmentation is highly valued and exclusive

Prevailing Narratives in Computer Science

- "Boring, repetitive work" should be automated or augmented for efficiency and profit
- **"Boring, repetitive work" is "robot work"**
- Augmentation is highly valued and exclusive

Historic Robots

- *Robot:* (Czech) compulsory service
  - Slav *robota*: servitude, hardship
- Robots: tool to replace “unskilled” work, servants



The robots are coming! When they do, you'll command a host of push-button servants.

By O. O. Binder

You'll Own "Slaves" by 1965

IN 1863, Abe Lincoln freed the slaves. But by 1965, slavery will be back! We'll all have personal slaves again, only this time we won't fight a Civil War over them. Slavery will be here to stay.

Don't be alarmed. We mean robot "slaves." Let's take a peek into the future to see what the Robot Age will bring. It is a morning of 1965. . .

You are gently awakened by soft chimes from your robot clock, which also turns up the heat, switches on radio news and signals your robot valet, whom you've affectionately named "Jingles." He turns on your shower, dries you with a blast of warm air, and runs an electric shaver over your stubble. Jingles helps you dress, tying your necktie perfectly and parting your hair within a millimeter of where you like it.

Down in the kitchen, Steela, the robot cook, opens a door in her own alloy body and withdraws eggs, toast and coffee from her built-in stove. Then she dumps the dishes back in and you hear her internal dishwasher bubbling as you leave for the garage.

In your robot car you simply set a dial for your destination and relax. Your automatic auto does the rest—following a radar beam downtown, passing other cars, slowing down in speed zones, gently applying radar brakes when necessary, even gassing up when your tank is empty. You give a friendly wave to robot traffic cops who break up all traffic jams with electronic speed and perception. Suddenly you hear gun shots. A thief is emptying his gun at a robot cop, who just keeps coming, bullets bouncing

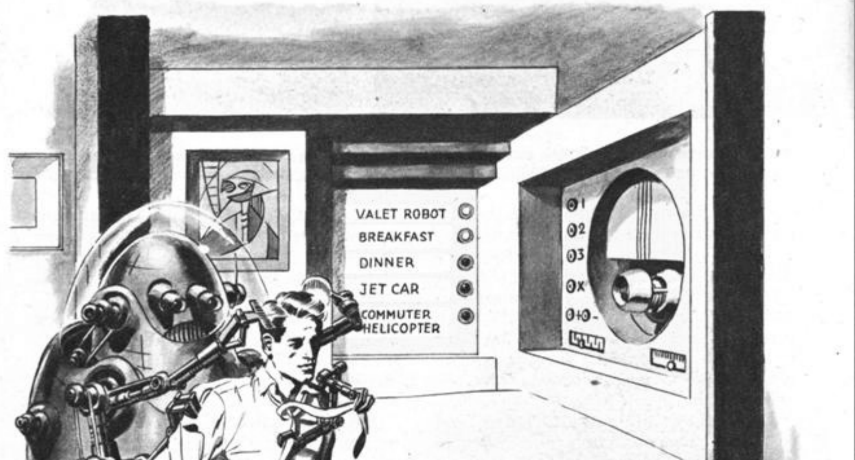
course the 1965 robots can be adjusted as you wish by buttons for high, average or low skill.

When a heavy snow falls you don't have to shovel the walk. Neither does your robot caretaker. He merely sprays cheap atomic heat around the grounds, melting the snow as fast as it falls. Yours is a robot home, too, turning all day on a foundation turntable to enjoy the utmost benefits of the sun.

At bedtime, you snap on the robot guard who detects any burglars electronically. It's a cheaper version of the robot alarm system in 1956, guarding precious documents like the original Constitution, in the National Archives Building.

During the night, no mice or rats can escape the super-sensitive ears and infra-red eyes of your roving robot cat. Back in 1956 scientists experimented with the first robot animals, such as the robot mole that could follow light beams, the robot moth dancing around flames and robot mice finding their way out of mazes.

Fanciful, this picture of the near future? A foretaste of such robot wonders



Robots will dress you, comb your hair and serve meals in a jiffy.

Metal star of Zombies Of The Stratosphere heeds his masters in science-fiction movie.

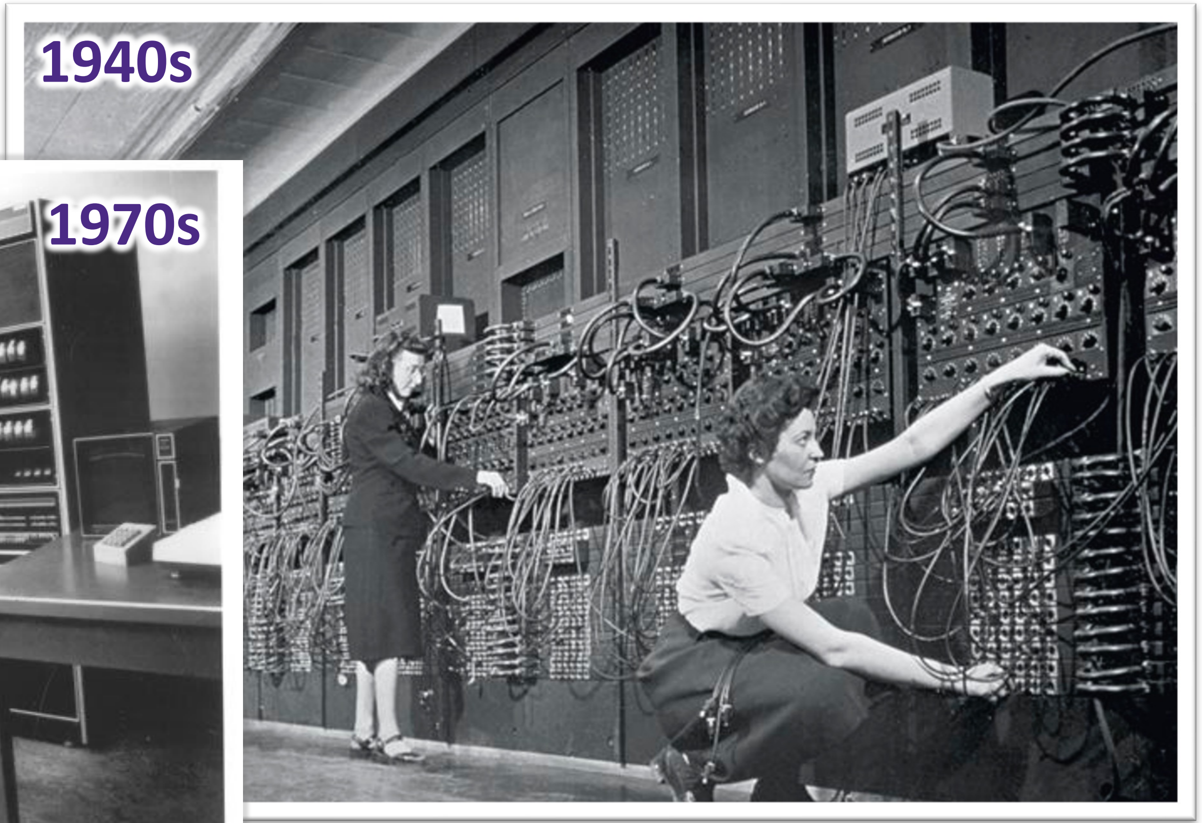
Prevailing Narratives in Computer Science

- "Boring, repetitive work" should be automated or augmented for efficiency and profit
- **"Boring, repetitive work" is "robot work"**
  - Performed by those deemed *less than human*
  - Robot work should be done by robots (non-human)
    - *"Robot work": anything unvalued by those with systemic power*
  - If the task can't be automated, use people (less-human)
    - Frequently, this ends up being marginalized people, who later have their jobs automated
- Augmentation is highly valued and exclusive

Prevailing Narratives in Computer Science

- "Boring, repetitive work" should be automated or augmented for efficiency and profit
- "Boring, repetitive work" is "robot work"
  - Performed by those deemed *less than human*
  - Robot work should be done by robots (non-human)
    - *"Robot work": anything unvalued by the powerful*
  - If the task can't be automated, use people (less-human)
    - Frequently, this ends up being marginalized people, who later have their jobs automated
- **Augmentation is highly valued and exclusive**

Programming, historically



https://s-media-cache-ak0.pinimg.com/564x/91/37/23/91372375e2e6517f8af128aab655e3b4.jpg

Jean Jennings (left), Marlyn Wescoff (center), and Ruth Lichterman program ENIAC at the University of Pennsylvania, circa 1946.
Photo: Corbis
http://fortune.com/2014/09/18/walter-isaacson-the-women-of-eniac/

Oh god, how did we get here?

How's your love life?

A little dull around the edges? Routine? Predictable? Boring? Maybe all it needs is a little Interlude. Interlude is the most stimulating computer game ever conceived. It combines a computer interview, an innovative programming concept, and a one-of-a-kind manual to turn your love life into exciting, adventurous, delicious fun!

Interlude is: romantic . . . playful . . . outrageous . . . a fantasy. Interlude is: ■ Wet fun on a hot summer night. (Interlude #21) ■ A surprise on the way home from dinner. (Interlude #42) ■ A bubble bath that ends with a bang. (Interlude #78) ■ An evening to rest while she does all the "work." (Interlude #25) ■ The most romantic of evenings. (Interlude #84) ■ A new twist to an old subject. (Interlude #69) ■ Just watching her . . . (Interlude #57) ■ An erotic fantasy! (Interlude #33)

With over 100 Interludes, you can satisfy all levels of interest and desire. Each Interlude is fully described in the manual, and the more elaborate ones are detailed with regard to settings, props, and mood-enhancing techniques. But we've saved a few super Interludes for that very special time when your interview indicates you're ready! At that time, you will be introduced to one of several Interludes held secret within the computer. (When you learn secret Interlude #99, your love life may never be the same again!) Interlude can give you experiences you'll never forget. Are you ready for it?

Interlude™
The Ultimate Experience.

Interlude, 10428 Westpark, Houston, Texas 77042 I'm really ready! Rush me_____copies of Interlude today. ✓235
☐ For the Apple II (16K) # ☐ For the TRS-80 (Level II-16K) ## ☐ $14.95 for cassette ☐ $17.95 for diskette.
Add $1.50 for shipping. Texas residents add 6% sales tax. My check (payable to Interlude) is enclosed. ☐
*Charge my ☐ MASTERCHARGE ☐ VISA account.
Account No._____ Expiration date_____
All charge customers must sign. Signature_____
Name_____ Age_____
Address_____
City_____ State_____ Zip_____
*CHARGE CUSTOMERS: Order by phone toll-free! **1-800-327-9009 Ext. 306** .Florida-1-800-432-7999 Ext. 306
Apple II is a registered trademark of Apple Computers, Inc. ## TRS-80 is a registered trademark of Radio Shack, a Tandy Co.

For a truly educational blog on this retro game, see here!

Modern Robots: Personal Computers

Prevailing Narratives in Computer Science

- "Boring, repetitive work" should be automated or augmented for efficiency and profit
- "Boring, repetitive work" is "robot work"
  - Performed by those deemed *less than human*
  - Robot work should be done by robots (non-human)
    - *"Robot work": anything unvalued by the powerful*
  - If the task can't be automated, use people (less-human)
    - Frequently, this ends up being marginalized people, who later have their jobs automated
- **Augmentation is highly valued and exclusive**

Automation – it’s complicated

- I don’t mean to vilify automation indiscriminately
  - Adaptive cruise control, autopilot, medical devices
- **However**, we need to consider the values that inform whether specific tasks should be automated
  - *Why* should this task be automated?
    - 737 MAX with MCAS – Boeing wanted to save money
  - *Who* does this automation seek to benefit?
    - Self driving cars replacing rideshare and taxi drivers