تاریخچه و نسل های کامپیوتر

با توجه به وابستگی میان سیستم عامل مورد استفاده در معماری کامپیوتر نسل های مختلف، در این بخش تاریخچه ای از کامپیوتر و سیستم عامل بطور همزمان ارائه می گردد. اولین کامپیوتر رقمی واقعی توسط یک ریاضیدان انگلیسی به نام چارلز بابیج (1871–1792**) طراحی شده است. اگر چه بابیج بیشتر عمر و ثروت خود را برای ساختن موتور تحلیلی صرف کرد، اما وی به نتیجه دلخواه و مطلوب خود نرسید. زیرا او فقط از مکانیک محض استفاده می کرد و فن‌آوری آن دوران امکان تولید چرخها و چرخ دنده هایی با دقت و ظرافت مورد نیاز وی را نداشت(نیاز به توضیح ندارد که موتور تحلیلی وی سیستم عامل نداشت).**

یک مطلب جالب تاریخی این است که بابیج فهمیده بود که موتور تحلیلی اش به نرم افزار نیاز دارد. بنابراین او خانم جوانی را به نام آدا استخدام نمود. او که دختر شاعر مشهور انگلیسی، لرد بایرون بود، به عنوان اولین برنامه نویس جهان شناخته شد. زبان برنامه نویسی آدا بعداً به نام او نامگذاریدگردید.

نسل اول (55–1945): لامپهای خلاء و تخته مدارهای سوراخدار

پس از تلاش ناموفق بابیج پیشرفت کمی در ساخت کامپیوترهای رقمی تا جنگ جهانی دوم صورت گرفت. در طی سالهای اواسط دهه 1940**، هوارد آیکن در دانشگاه هاروارد، جان**

فون نیومن در مؤسسه مطالعات پیشرفته پرینستون، ج. پرسپراکرت و ویلیام ماکلی در دانشگاه پنسیلوانیا و کنراد زاس در آلمان به همراه دیگران در ساخت موتورهای محاسباتی با استفاده از لامپهای خلاء موفق شدند. اندازه این ماشین ها بسیار بزرگ بود و به همراه دهها هزار لامپ خلاء داخل اتاق ها را پر میکرد، اما از ارزانترین کامپیوترهای شخصی امروزی کندتر کار می کرد.

در روزهای اولیه، یک گروه از مردم تمامی مراحل طراحی، ساخت، برنامه نویسی، استفاده و نگهداری از یک ماشین را به عهده داشتند. کلیه برنامه ها به طور مطلق به زبان ماشین نوشته می شد و اغلب به وسیله سیم بندی تخته مدارهای سوراخدار و به منظور کنترل عملیات پایه ای ماشین انجام می شد. زبانهای برنامه نویسی شناخته نشده بود (حتی زبان اسمبلی) و هیچ کس نامی از سیستم عامل نشنیده بود. روند معمول کار به این صورت بود که برنامه نویسان برای رزرو وقت برای کار با ماشین مراجعه کرده و بر روی برگه نام نویسی برای یک دوره معین زمانی ثبت نام می کردند. سپس در وقت تعیین شده به اتاق ماشین وارد می شدند و تخته مدار سوراخدار خود را در ماشین قرار میدادند و ساعاتی را منتظر می ماندند، بدان امید که هیچیک از 20000 **(یا چند) لامپ خلاء، در طی اجرای برنامه نسوزد. اکثر برنامه ها، محاسبات عددی معمولی مانند تهیه جداول سینوس و کسینوس بود.**

در اوایل دهه 1950 **روند عملیات با ظهور کارت های منگنه (پانچ) بهبود یافت. حال به جای تخته مدارهای سوراخدار، برنامه ها بر روی کارت ها منگنه می شد و کامپیوترها توسط دستگاه کارتخوان برنامه ها را می خواندند. سایر کارها مانند همان روال قبل بود.**

نسل دوم (65–1955) ترانزیستور و سیستم های دسته ای

ظهور ترانزیستور در اواسط دهه 1950 **یک انقلاب بنیادین بود. قابلیت اطمینان کامپیوترها به اندازه کافی بالا رفت، بطوریکه سازندگان می توانستند کامپیوترها را تولید کرده و به مشتری ها بفروشند. زیرا طول عمر آنها جهت استفاده مفید برای انجام کارهای کاربران، رضایت بخش بود. برای اولین بار به وضوح مشاهده می شد که متخصصین طراحی، ساخت، برنامه نویسی، تعمیر و نگهداری و اپراتوری از یکدیگر تفکیک شده اند. این ماشین ها در اتاقهای مخصوص کامپیوتر با تهویه مطبوع مستقر می شد و گروهی از اپراتورهای حرفه ای راهبری آنرا به عهده می گرفتند. فقط شرکت های بزرگ، ادارات دولتی و یا دانشگاهها از عهده پرداخت هزینه چند میلیون دلاری این ماشین ها بر می آمدند. برای اجرای یک کار (مثلا یک برنامه یا یک مجموعه از برنامه ها)، یک برنامه نویس برنامه خود را بر روی کاغذ می نوشت (به زبان فرترن یا اسمبلی)، سپس آنرا بر روی کارت ها منگنه می کرد و یک دسته کارت را به اتاق کامپیوتر می آورد و به یکی از اپراتورها تحویل میداد.**

وقتی که کامپیوتر کار در حال اجرا را به اتمام می رساند، یک اپراتور به سمت چاپگر می رفت و کاغذهای خروجی برنامه را جدا و به اتاق خروجی منتقل می کرد تا بعداً برنامه نویس آنها را بردارد. سپس او یکی دیگر از دسته کارتها را که از اتاق ورودی آورده بود، در دستگاه کارتخوان قرار میداد و در صورتیکه برنامه به مترجم فرترن نیاز داشت، وی دسته کارت مترجم فرترن را نیز از داخل کمد فایل ها برداشته و در دستگاه کارتخوان قرار میداد. هنگامیکه اپراتورها برای انجام این کارها در اتاق ماشین راه می رفتند، مدت زیادی از وقت کامپیوتر تلف می شد.

به خاطر هزینه گزاف تجهیزات، تعجبی نداشت که مردم به سرعت به دنبال روشهایی برای کاستن زمان تلف شده باشند. راه حلی که به طور کلی پذیرفته شد، سیستم های دسته ای بود. ایده پشت این روش این بود که یک سبد پر از دسته کارتها در اتاق ورودی جمع آوری شود، سپس کلیه آنها به وسیله دستگاه کارتخوان یک کامپیوتر کوچک و نسبتاً ارزان مانند IBM 1401 **خوانده شده و از طریق یک نوار گردان بر روی یک نوار مغناطیسی ذخیره گردد. این کامپیوترها برای خواندن کارتها، کار با نوار گردان ها و چاپ خروجی ها کارایی خوبی داشتند، اما برای محاسبات عددی مناسب نبودند. از طرف دیگر یک ماشین گرانقیمت ، مانند** IBM 7094 **برای پردازش و محاسبات واقعی استفاده می شد. این وضعیت در (شکل** 2–1**) نشان داده شده است. پس از حدود یک ساعت که برای جمع**

آوری یک دسته از کارها بر روی نوار صرف می شد، یک اپراتور نوار را از اتاق ورودی برداشته و به اتاق ماشین منتقل می کرد و در آنجا در یک نوار گردان قرار میداد. سپس یک برنامه مخصوص (جد سیستمهای عامل امروزی) را بار می کرد تا اولین کار را از روی نوار بخواند و اجرا نماید. خروجی به جای چاپگر بر روی یک نوار دیگر نوشته می شد. پس از اتمام هر کار، سیستم عامل به صورت خودکار کار بعدی را از نوار می خواند و شروع به اجرای آن می کرد. وقتی که تمامی دسته کارها اجرا می شد، اپراتور نوارهای ورودی و خروجی را برمی داشت، نوار ورودی را با دسته بعدی جایگزین کرده، نوار خروجی را به ماشین 1401 **منتقل می کرد تا عملیات چاپ به صورت ناپیوسته (به کامپیوتر اصلی متصل نیست) انجام شود.**

ساختار یک کار ورودی معمولی در (شکل 3−1**) نشان داده شده است. این ساختار با یک کارت** $JOB **شروع می شود که مشخص کننده حداکثر زمان اجرا به دقیقه، شماره حساب برای حسابداری هزینه ها و نام برنامه نویس می باشد و سپس یک کارت** $FORTRAN **قرار دارد که به سیستم عامل اعلام میکند که مترجم فرترن را از روی نوار سیستم بار نماید. سپس کارتهای برنامه ای که باید ترجمه شود قرار دارد و کارت** $LOAD **به دنبال آن قرار گرفته است. این کارت سیستم عامل را وادار می کند که برنامه** Object **(مستقیماً ترجمه شده) را بار نماید. (برنامه ترجمه شده اغلب بر روی یک نوار چرکنویس (موقتی)**

نوشته می شود و باید برای اجرای دوباره بار شود.) سپس نوبت به کارت $RUN **می رسد که به سیستم عامل می گوید اجرای برنامه ها را به همراه داده هایش که در کارتهای بعدی قرار دارد، آغاز نماید. در انتها کارت** $END **قرار گرفته است که نشان دهنده پایان کار است. این کارت های کنترلی قدیمی جد زبانهای کنترل کار پیشرفته و مفسرهای فرمان جدید بودند.**

کامپیوترهای بزرگ نسل دوم بیشتر برای محاسبات مهندسی و علمی مانند حل معادلات مشتق جزیی به کار می رفت. برنامه آنها بیشتر به زبان فرترن و اسمبلی نوشته می شد و معمولاً از سیستمهای عامل FMS **و** IBSYS **(سیستم عامل** IBM **برای** 7094**) استفاده می شد.**

نسل سوم (1980 – 1965): مدارات مجتمع و چند برنامگی

در اوایل دهه 1960**، اکثر سازندگان کامپیوتر دو خط تولید مجزا و کاملاً ناسازگار داشتند. در یک طرف کامپیوترهای کلمه گرا با مقیاس بزرگ مانند** 7094 **قرار داشت که بر ای محاسبات عددی در مهندسی و علوم به کار می رفت. در طرف دیگر کامپیوترهای تجارتی نویسه گرا مانند** 1401 **قرار داشت که به طور گسترده برای ذخیره سازی نوارها و چاپ در بانکها و مؤسسات بیمه به کار می‌رفت.**

توسعه و نگهداری دو خط تولید کاملاً متفاوت، برای سازندگان پر هزینه بود. به علاوه، بسیاری از مشتری های جدید کامپیوتر به یک ماشین کوچک نیاز داشتند، اما بعداً کار خومد را توسعه داده و در خواست یک ماشین بزرگتر می کردند که همه برنامه های قدیمی آنها را نیز با سرعت بیشتر اجرا نماید. IBM **تلاش کرد تا هر دو مشکل را یکجا حل نماید و این کار را با ارائه سیستم** 360 **انجام داد.** 360 **یکسری ماشین بود که از نظر نرم افزاری با یکدیگر سازگار بودند. در این سری، طیف وسیعی از ماشین ها (از اندازه** 1401 **تا خیلی قوی تر از** 7094**) وجود داشت. تفاوت ماشین ها فقط در قیمت و کارایی (حداکثر اندازه حافظه، سرعت پردازنده، تعداد دستگاههای** I/O **مجاز و نظیر آن) آنها بود. از آنجا که همه این ماشین ها معماری و مجموعه دستورالعمل ها یکسانی داشتند، برنامه های نوشته شده در یکی از آنها در سایر ماشین ها نیز (حداقل از نظر تئوری) قابل اجرا بود. دیگر اینکه** 360 **برای انجام محاسبات در دو زمینه علمی و تجارتی ساخته شده بود. بنابراین یک خانواده از ماشین ها نیاز، تمامی مشتریان را برآورده می ساخت. در سالهای، بعد** IBM **جانشین های سازگار با خط تو.لید** 360**، اما با فن آوری پیشرفته تر، سری های** 370**،** 4300**،** 3080**،** 3090 **را ارائه داد.**

سری 360 **اولین خط تولید کامپیوترهای بزرگ بود که از مدارت مجتمع با مقیاس کوچک استفاده کرد. بنابراین نسبت به کامپیوترهای نسل دوم که از ترانزیستورهای جداگانه**

ساخته می شد، نسبت به ک ارایی از هزینه بالاتری برخوردار بود. این یک موفقیت بود و ایده خانواده کامپیوترهای سازگار خیلی زود توسط دیگر سازندگان بزرگ به کار گرفته شد. نوادگان این ماشین ها هنوز هم در برخی از سایت های کامپیوتری به طور پراکنده در حال استفاده می باشند، اما استفاده از آنها به سرعت رو به انحطاط است.

بزرگترین مایه قوت ایده "یک خانواده"، با بزرگترین ضعف آن نیز همراه بود. مشکل از آنجا سرچشمه می گرفت که کلیه نرم افزارها، از جمله سیستم عامل، الزاماً باید بر روی انواع کامپیوترهای خانواده کار می کرد. این برنامه باید بر روی کامپیوترهای کوچکی که جایگزین 1401 **شده بود و برای نسخه برداری از کارتها بر روی نوار به کار می رفت، اجرا می شد. همین برنامه باید بر روی سیستم های خیلی بزرگ که جایگزین** 7094 **شدده بود و برای محاسبات خیلی سنگین مثل پیش بینی هواشناسی به کار می رفت، نیز اجرا می شد. همچنین باید بر روی کامپیوترهایی که تعداد لوازم جانبی آنها متفاوت (کم یا زیاد) بود، به خوبی کار می کرد. همین برنامه مجبور بود در محیطهای تجارتی و علمی مورد استفاده قرارگیرد. بالاتر از همه اینها، برنامه باید در کاربردهای گوناگون از کارایی بالایی برخوردار می بود.**

هیچ راهی وجود نداشت که IBM **(یا هرکس دیگر) بتواند یک تکه برنامه بنویسد که همه این نیازهای ناسازگار را برآورده سازد. نتیجه کار یک سیستم عامل عظیم الجثه و بسیار**

پیچیده خواهد بود که شاید صدها برابر بزرگتر از FMS **باشد. این سیستم عامل باید از میلیونها خط برنامه اسمبلی تشکیل شده باشد که توسط هزاران برنامه نویس نوشته شده است. همچنین شامل هزاران هزار اشکال خواهد بود که برای برطرف کردن آنها ناگزیر به ارائه جریان دائمی از نسخه های پی دوپی برنامه خواهیم شد. هر نسخه جدید چند اشکال را برطرف کرده و چند اشکال جدید به آن می افزاید و تعداد اشکالات برنامه ثابت می ماند. یکی از طراحان** OS/360 **، فرد بروکس پس از آن کتابی نافذ با کنایه و شوخی نوشت (بروکس،** 1975**) و تجربیاتش را با** OS/360 **توصیف کرد. از آنجا که ارائه خلاصه ای از آن کتاب در اینجا غیرممکن است، همین کافی است که بگوییم طرح روی جلد آن یک گله از جانوران ما قبل تاریخ را نشان می دهد که در یک گودال پر از قیر فرو رفته اند. جلد کتاب گالوین و سیلبرز چتر (**1994**) نیز بیانگر نکته ای مشابه آن است.** OS/360 **و دیگر سیستمهای عامل مشابه نسل سوم، با وجود بزرگی اندازه و مشکلاتشان بیشتر مشتری های خود را راضی نگه داشتند. همچنین این سیستمهای عامل چندین تکنیک کلیدی جدید را که در سیستمهای عامل نسل دوم وجود نداشت، متداول نمودند. شاید مهمترین این تکنیک ها چند برنامگی بود. در** 7094 **وقتی که کار جاری برای تکمیل یک عملیات** I/O **مثل نوار گردان منتظر می شود، واضح است که** CPU **بیکار می شود تا عملیات** I/O **به اتمام رسد. در محاسبات علمی سنگین با تنگنای محاسباتی ،** I/O **به ندرت به کار می رود و در**

نتیجه زمان تلف شده اهمیت چندانی نخواهد داشت. اما در پردازش داده های تجارتی ، زمان انتظار I/O **اغلب** 80 **تا** 90 **درصد کل زمان را به خود اختصاص می دهد، بنابراین باید برای پرهیز از این همه بیکاری پردازنده، چاره‌ای بجوییم. راه حل ارائه شده این بود که حافظه را به چند تکه تقسیم بندی نماییم و همانطور که در (شکل** 4–1**) دیده می شود، یک کار مجزا را در هر پخش قرار دهیم.**

وقتی که یک کار برای تکمیل عملیات I/O **منتظر می ماند، کار دیگری پردازنده را در اختیار می‌گیرد. اگر تعداد کارهای موجود در حافظه کافی باشد، می توان پردازنده را تقریباً در صد درصد زمانها مشغول نگه داشت. نگهداری همزمان چند کار در درون حافظه، نیاز به سخت افزار مخصوص جهت محافظت از هر کار در برابر دسترسی پنهانی و شیطنت آمیز دیگر کارها دارد. اما** 360 **و دیگر سیستم های نسل سوم به این سخت افزار مجهز بودند.**

دیگر ویژگی بزرگ سیستمهای عامل نسل سوم این بود که بلافاصله پس از ورود کارها به اتاق کامپیوتر، می توانستند کارتها را خوانده و به دیسک منتقل نمایند. بنابراین هرگاه یک کار در حال اجرا به پایان می رسد، سیستم عامل می تواند یک کار جدید را از روی دیسک برداشته و در یک بخش خالی شده از حافظه بار نماید و سپس آنرا به اجرا در آورد. این تکنیک که Spooling **(عملیات پیوسته همزمان دستگاههای جانبی) نامیده شد، برای**

خروجی نیز به کار گرفته شد. با Spooling **دیگر نیازی به** 1401 **ها ونوارگردان های اضافی و حمل نوارها نبود.**

اگر چه سیستمهای عامل نسل سوم برای محاسبات علمی بزرگ و پردازش کلان داده های تجارتی مناسب بود، آنها بطور کلی هنوز سیستم های دسته ای بودند. بعضی از کاربران حسرت روزهای نسل اول کامپیوترها را به دل داشتند. روزهایی که برای چندین ساعت همه امکانات ماشین در اختیار آنها بود و می توانستند به سرعت برنامه هایشان را اشکالزدایی نمایند. در سیستم های نسل سوم زمان بین تحویل برنامه به اتاق ورودی و دریافت خروجی اغلب چندین ساعت به طول می انجامید. بنابراین تنها جابجایی یک کاما می توانست منجربه خطای ترجمه شده و وقت برنامه نویس را نیم روز دیگر تلف نماید.

آرزوی داشتن زمان پاسخ کوتاه، راه را برای سیستم های اشتراک زمانی هموار ساخت. در سیستم های اشتراک زمانی (چند برنامگی دگرگون شده) هر کاربر یک ترمینال بر خط (پیوسته) در اختیار دارد. حال اگر 20 **کاربر وارد سیستم شده باشند و** 17 **نفر از آنها در حال فکر کردن، صحبت کردن، یا نوشیدن قهوه باشند، پردازنده به آن سه کار باقیمانده تخصیص خواهد یافت که منتظر سرویس می باشند. کاربرانی که در حال اشکالزدایی برنامه هایشان هستند از فرامین کوتاه (مثل ترجمه یک تکه برنامه پنج صفحه ای) استفاده**

می نمایند (در مقابل فرامین طولانی مانند مرتب سازی یک فایل یک میلیون رکوردی). در نتیجه کامپیوتر قادر است که به تعدادی از کاربران در کنار یکدیگر سرویس مجاوره ای سریع ارائه نماید. حتی می تواند در مواقعی که cpu **بیکار می ماند، کارهای دسته ای بزرگ را در پشت صحنه به اجرا در آورد.** CTSS **اولین سیستم اشتراک زمانی جدی بود که دانشگاه** M.I.T. **آنرا با ایجاد تغییرات خاصی در** 7094 **ارائه داد (کورباتو و دیگران،** 1962**). اما استفاده از سیستم های اشتراک زمانی عملا فراگیر نشد تا اینکه استفاده از سخت افزار لازم برای حفاظت در کامپیوترهای نسل سوم رایج شد.**

پس از موفقیت سیستم CTSS**، دانشگاه** M.I.T. **و** Bell Labs **و** General Electric **تصمیم گرفتند که طراحی و پیاده سازی یک ماشین اشتراک زمانی را شروع کنند. آنها این ماشین را که می توانست صدها کاربر را به طور همزمان تحت پوشش اشتراک زمانی قرار دهد، "صنعت همگانی کامپیوتر " نامیدند. مدل آنها برای اینکار سیستم توزیع الکتریسته بود. قوتی که شما به توان الکتریکی نیاز دارید، یک دو شاخه را در پریز روی دیوار قرار می دهید و به هر اندازه که نیاز دارید از توان الکتریکی استفاده می نمایید. طراحان این سیستم که به** MULTICS **(خدمات محاسباتی و اطلاعاتی تسهیم شده) معروف شد، رویای یک ماشین بزرگ را در سر می پروراندند که بتواند توان محاسباتی مورد نیاز همه اهالی بستون را فراهم آورد. این موضوع که سی سال بعد ماشین هایی بسیار قوی تر از** GE-645

به عنوان کامپیوترهای شخصی، فقط به قیمت ناچیز چند هزار دلار فروخته شود، در آن زمان یک افسانه علمی محض بود. برای کوتاه کردن این داستان بلند، MULTICS **شروع به ارائه ایده های اولیه در نشریات کامپیوتری کرد، اما ساخت آن بسیار مشکل تر از حد انتظار بود.** Bell Labs **از این پروژه کنار کشید.** General Electric **نیز، هم این پروژه و هم تجارت کامپیوتر را رها کرد. عاقبت،** MULTICS **جهت استفاده در یک محیط تولیدی در** M.I.T. **و چندین سایت کامپیوتری دیگر به شکل نسبتاً خوبی به اجرا در آمد . اما مفهوم صنعت همگانی کامپیوتر متروک ماند تا اینکه قیمت کامپیوترها به حد نازلی برسد. هنوز هم** MULTICS **نفوذ زیادی بر روی سیستم های بعد از خود دارد که شرح آن در چند کتاب آمده است. (کورباتو و دیگران،** 1972 – **کورباتو ویسوتسکی،** 1965 – **دالی و دنیس،** 1968 – **ارگانیک ،** 1972 – **سالترز ،** 1974**)پیشرفت بزرگ دیگر، طی سالهای نسل سوم، پدیده رشد کامپیوترهای کوچک بود که با** DEC PDP-1 **در سال** 1961 **شروع شدد.** POP-1 **فقط چهار هزار کلمه** 18 **بیتی داشت، اما با قیمت** 120000 **دلار برای هر ماشین (کمتر از** 5٪ **قیمت** 7094**) عرضه شد و به راحتی به فروش رفت. برای برخی از کارهای غیرعددی مشخص، سرعت آن در حدود** 7094 **بود و تولد صنعت کاملاً جدیدی را موجب شد. به سرعت سری جدیدی از** POP **ها (برعکس خانواده** IBM**، همگی ناسازگار بودند) عرضه شد که در** PDP-11 **به اوج خود رسید.**

یکی از دانشمندان کامپیوتر در Bell Labs **که در پروژه** MULTICS **کار کرده بود، به نام کن تامپسون، مدتی بعد یک کامپیوتر** PDP-7 **پیدا کرد که کسی از آن استفاده نمی کرد. او شروع به نوشتن یک نسخه تک کاربرد از** MULTICS**کرد. این کار بعداً توسعه پیدا کرد و منجربه پیدایش سیستم عامل** UNIX **شد. این سیستم عامل در محیط های علمی و دانشگاهی، ادارات دولتی و بعضی از شرکت ها به وفور مورد استفاده قرار گرفت.**

تاریخچه UNIX **در جای دیگری (مثل سالوس،** 1994**) گفته شده است. فقط کافی است که بگوییم به علت در دسترس بودن کد برنامه این سیستم عامل، سازمانهای مختلف نسخه های گوناگون و ناسازگاری از آنرا برای خودشان نوشتند و این کار منجربه هرج و مرج شد. به منظور اینکه بتوان برنامه ای نوشت که بر روی انواع سیستم های** UNIX **قابل اجرا باشد،** IEEE **یک استاندارد برای** UNIX **به نام** POSIX **بنا نهاد. اکنون بیشتر نسخه های** UNIX **از آن استاندارد پیروی می کند.** POSIX **حداقل واسط فراخوان سیستمی را تعریف می کند که سیستم های** UNIX **سازگار باید آنرا پشتیبانی نمایند. در حقیقت ، خیلی از سیستمهای عامل کنونی نیز از واسط** POSIX **پشتیبانی می نمایند.**

نسل چهارم (1980 تاکنون): کامپیوترهای شخصی

با توسعه مدارات مجتمع با مقیاس بزرگ (LSI)**، که تراشه هایی شامل هزاران ترانزیستور در یک سانتیمتر مربع از سیلیکیون بود، کامپیوترهای شخصی** (PC) **پا به عرصه وجود نهاد.**

از نقطه نظر معماری کامپیوترهای شخصی تفاوت چندانی با کامپیوترهای کوچک گروه PDP-11 **نداشت، اما از نظر قیمت تفاوت عمده ای بین آنها بود. همانطور که ظهور کامپیوترهای کوچک موجب شد که هر دانشگاه یا شرکتی بتواند برای خود یک کامپیوتر تهیه نماید، ظهور تراشه های ریز پردازنده نیز موجب شد که هر فرد بتواند برای خود یک کامپیوتر شخصی و اختصاصی تهیه نماید. کامپیوترهای شخصی یا بیشترین توان عملیاتی، در مراکز تجارتی و دانشگاهی استفاده می شود و از آنها به نام ایستگاه های کاری یاد می شود. معمولا این ایستگاه های کاری از طریق یک شبکه به یکدیگر متصل شده اند.**

گسترش کامپیوترهای شخصی موجب شد که این کامپیوترها در سطح وسیعی در دسترس همگان قرار گیرد. این امر به همراه گرافیک عالی و محیط های مجاوره ای قوی منجربه رشد یک صنعت بزرگ برای تولید نرم افزارهای کامپیوتری شخصی شد. استفاده کردن از بیشتر این نرم افزارها آسان بود، بدین معنی که کاربران نه تنها نیاز به داشتن دانش کامپیوتری نداشتند، بلکه حتی استفاده از آنها به آموزش چندانی هم نیاز نداشت. در واقع نسبت به OS/360 **یک تحول اساسی رخ داده بود، چرا که آن سیستم عامل آنقدر پر رمزو راز بود که یک کتاب بزرگ برای آن نوشته شده بود. (کادو،** 1970**).**

در ابتدا دو سیستم عامل در صحنه کامپیوترهای شخصی و ایستگاه های کاری حکمفرمایی می‌کردند: Microsofts MS-DOS UNIX **،** MS-DOS **به طور گسترده بر روی**

کامپیوترهای شخصی IBM **و دیگر ماشین هایی که از پردازند** Intel 8080 **و فرزندان آن استفاده می کردند، مورد استفاده قرار گرفت. پردازنده های بعدی این خانواده** 80286،80386،80486 **(که از آنها به نامهای** 286،386،486 **نیز یاد می شود) و سپس پنتیوم و پنتیوم پرو بودند. اگر چه نسخه اولیه** MS-DOS **تقریباً به صورت ابتدایی نوشته شده بود، اما نسخه های بعدی از ویژگی های پیشرفته تری برخوردار بودند و بسیاری از آنها از** UNIX **الهام گرفته شده بود.** WINDOWS **که جانشین** MS-DOS **بود، ابتدا بر روی** MS-DOS **اجرا می شد (بیشتر شبیه یک پوسته عمل می کرد، تا یک سیستم عامل واقعی). اما در سال** 1990 **نسخه های مستقلی از** WINDOWS **پا به عرصه وجود نهاد که اولین آنها** WINDOWS 95 **نام داشت و دیگر نیاز به این نبود که** MS-DOS **آنرا پشتیبانی نماید. سیستم عامل دیگر این شرکت** WINDOWS NT **نام دارد که تا حدی سازگار با** WINDOWS 95 **است، اما از درون کاملاً بازنویسی شده است.**

رقیب بزرگ دیگر، UNIX **بود که در ایستگاههای کاری و کامپیوترهای خدمت گذار شبکه حکمفرمایی می کرد. مخصوصاً استفاده از آن در ماشین هایی که از تراشه های کارآمد** RISC **بهره می بردند، فراگیر شده اگر چه فقط یک کاربر از این ماشین ها استفاده می کرد، اما توان محاسباتی آنها معمولاً به اندازه کامپیوترهای کوچک بود. بنابراین منطقی**

بود که این ماشین ها مجهز به سیستم عاملی باشند که در اصل برای کامپیوترهای کوچک طراحی شده بود، یعنی UNIX **رشد شبکه های کامپیوترهای شخصی طی سالهای اواسط دهه** 1980 **پیشرفتی مهم و جالب توجه بود.**

در این شبکه ها سیستمهای عامل شبکه یا سیستمهای عامل توزیع شده اجرا می شد (تننباوم، 1995**). در یک سیستم عامل شبکه، کاربران از وجود ماشین های مختلف در شبکه با خبرند. آنها می توانند از دور وارد یک ماشین شوند و همچنین فایل های یک ماشین را بر روی ماشین دیگر نسخه برداری نمایند. هر ماشین سیستم عامل محلی خودش را اجرا می کند و کاربر یا کاربران محلی مخصوص به خود دارد.**

تفاوت اساسی بین سیستمهای عامل شبکه و سیستمهای عامل تک پردازنده وجود ندارد. واضح است که آنها به کنترل کننده های واسط شبکه ای و نرم افزاری سطح پایین برای گرداندن آن نیاز دارند، مانند برنامه هایی که برای ورود به سیستم از راه دور و دسترسی راه دور به فایل ها نوشته می شود، اما اضافه کردن این نرم افزارها ساختار اصلی سیستمهای عامل را تغییر نخواهد داد.

در مقابل، یک سیستم عامل توزیع شده ، خود ر ا مانند یک سیستم تک پردازنده ای قدیمی به کاربرانش نشان می دهد، علیرغم اینکه واقعاً اینگونه نیست و سیستم از چندین پردازنده تشکیل شده است. کاربران نباید از این امر آگاه شوند که برنامه آنها در کجا به اجرا در

می آید و یا فایلهای آنها در کجا قرار دارد. همه این امور باید توسط سیستم عامل به صورت خودکار و با کارآیی بالا انجام شود. (سیستم باید از دیدگاه کاربر شفاف باشد. هرچیز را با نام آن فراخوانی می کنیم و کاری به آدرس آن نداریم.)

سیستمهای عامل توزیع شده واقعی چیزی بیشتر از افزودن یک تکه برنامه به یک سیستم عامل تک پردازنده ای است، چرا که سیستم های متمرکز و توزیع شده در مسیرهای بحرانی با یکدیگر متفاوتند. به عنوان مثال، اغلب سیستم های توزیع شده به برنامه های کاربردی اجازه میدهند که در یک زمان بر روی چندین پردازنده (بطور موازی) به اجرا در آید، بنابراین به الگوریتم های پیچیده ای برای زمانبندی پردازنده نیازمند است تا اینکه بتواند پردازش موازی را بهینه سازد.

تأخیرهای انتقال در طی شبکه اغلب بدین معنی است که این (و دیگر) الگوریتم ها باید با اطلاعات ناقص، منسوخ یا حتی غلط اجرا شوند. این وضعیت با سیستم های تک پردازنده ای تفاوت اساسی دارد، چرا که در آنجا سیستم عامل اطلاعات کاملی راجع به وضعیت سیستم دارد.