ریزپردازنده ها

فصل سوم
میکروکنترلرها و سیستم های تعبیه شده

پاییز ۱۳۹۶

1

سیستم های تعبیه شده

- سیستم تعبیه شده به این معناست که پروسسور (پردازنده) و سایر امکانات در یک سیستم واحد گنجانده شده اند.

2

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

- استفاده از تراشه *ASIC*
- استفاده از قطعات گسسته
- استفاده از کامپیوترهای صنعتی
- استفاده از تراشه های برنامه پذیر از قبیل *CPLD* و *FPGA*
- استفاده از پردازنده ها

3

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از تراشه *ASIC*

- *ASIC* مخفف ***A****pplication* ***S****pecific* ***I****ntegrated* ***C****ircuit* است.
- در این روش کلیه نیازهای سیستم در قالب یک مدار مجتمع، توسط تکنولوژی VLSI و به صورت بهینه طراحی می شود.

عیب

- هزینه اولیه طراحی مهندسی بالا
- طی نمودن پروسه زمان بر و پرهزینه طراحی تراشه در زمان ارتقای سیستم

فقط در سیستم هایی با تولید بالا مانند گوشی های تلفن همراه یا کاربردهای نظامی که کارآیی و قابلیت اطمینان اهمیت دارد، استفاده می شود.

4

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از قطعات گسسته

- طراحی سیستم توسط المان هایی مانند ترانزیستورها و تراشه هایی مانند سری *74XX* انجام می شود.

- به دلیل تعداد المان های زیاد مورد استفاده، ابعاد و صرف توان سیستم بسیار افزایش می یابد ⟸ امروزه به ندرت از این روش استفاده می شود

5

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از کامپیوترهای صنعتی

- در برخی از کاربردها مانند دستگاه های خودپرداز *ATM* می توان از کامپیوترهای صنعتی به عنوان پردازنده استفاده کرد.

- سخت افزار سیستم به صورت آماده بوده و نیازی به طراحی ندارد.
- سیستم عامل های موجود از قبیل *Linux* و *Windows CE* کلیه سرویس های اصلی مورد نیاز سیستم را تأمین می کند.
- می توان از بسته های نرم افزاری متعددی برای نوشتن برنامه استفاده نمود.

6

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از کامپیوترهای صنعتی

معایب

- ابعاد بزرگ و مصرف توان بالای سیستم باعث می شود که در بسیاری از سیستم های کم مصرف و قابل حمل نتوان از این روش استفاده کرد.
- سیستم عامل های معمول مانند ویندوز قابلیت اطمینان بالایی ندارند و نمی توان در کاربردهای حساس از آن ها استفاده کرد.
- به صورت بلادرنگ (*Real Time*) عمل نمی کنند که همین امر، کاربرد آنها را محدود می کند.

7

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از تراشه های برنامه پذیر از قبیل *CPLD* و *FPGA*

- این ادوات در مقایسه با تراشه های *ASIC* هزینه طراحی اولیه کمتری دارند.
- کارد اصلی تراشه های *FPGA* در سیستم های پردازش اطلاعات موازی و پرسرعت است.
- از تراشه های *CPLD* بیشتر به عنوان کنترل کننده حافظه استفاده می شود.

عیب

- مصرف توان تراشه های *FPGA* بیش از میکروکنترلرها است

کمتر به عنوان کنترل کننده اصلی سیستم در ادوات تعبیه شده استفاده می شوند.

8

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از پردازنده ها

- با توجه به نوع کاربرد
  - میکروپروسسورهای عمومی (همه منظوره)
  - پردازنده های *DSP*
  - میکروکنترلرها

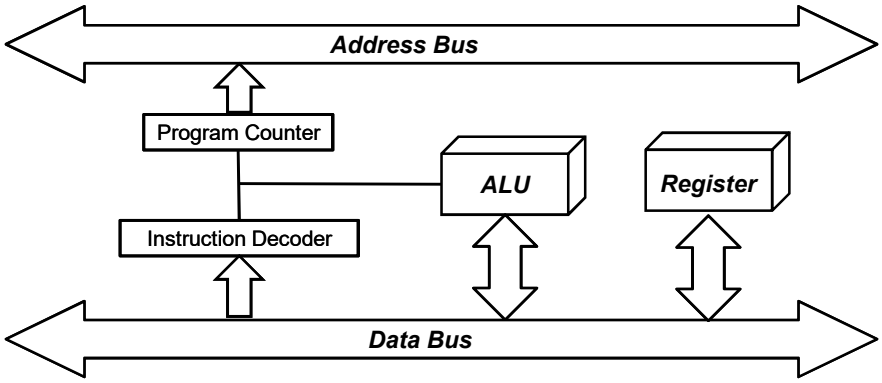
9

میکروپروسسورها

- نخستین میکروپروسسور در سال ۱۹۷۴ توسط شرکت *Intel* به بازار عرضه شد.

10

معماری میکروپروسسورها



11

معماری میکروپروسسورها

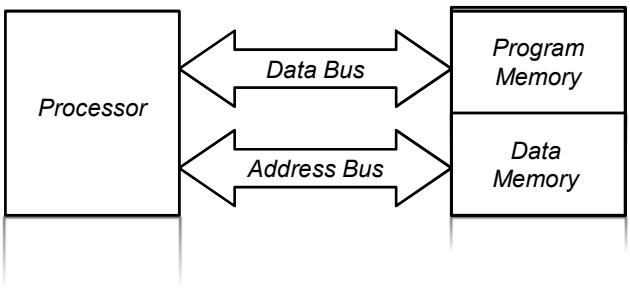
- در حالت کلی، پردازنده ها دارای ادوات داخلی یا ادوات ورودی ـ خروجی نمی باشند و از گذرگاه های داده و آدرس برای ارتباط با اجزای جانبی استفاده می کنند.

- دو معماری متفاوت برای دسترسی به حافظه
  - معماری *Von Neumann*
  - معماری *Harvard*

12

معماری *Von Neumann*

- گذرگاه آدرس و داده برای دسترسی به حافظه کد از قبیل *ROM*، *EEPROM* و *Flash* و حافظه داده (*RAM*) مشترک است.



13

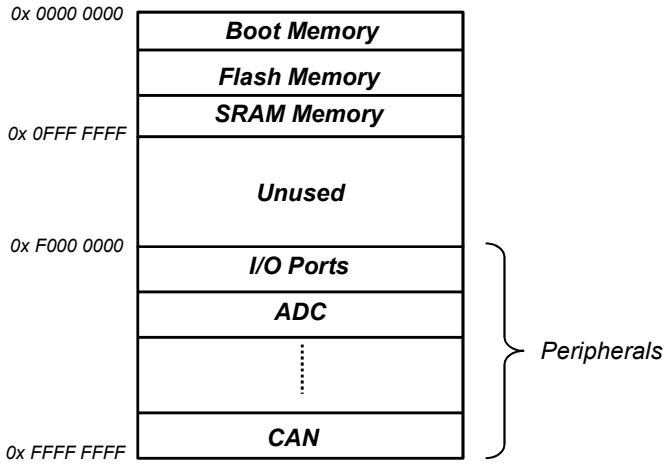
معماری *Von Neumann*

- سادگی و اقتصادی بودن

- با اجرای دستوراتی که با حافظه در ارتباط هستند، فراخوانی دستور بعد تا کامل شدن نوشتن یا خواندن از حافظه داده متوقف می شود و به این ترتیب سرعت اجرای دستورها کاهش می یابد.

14

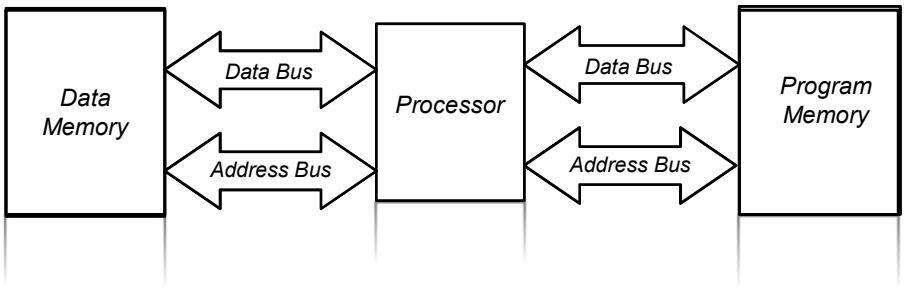
تقسیم بندی فضای حافظه در معماری *Von Neumann*



15

معماری *Harvard*

گذرگاه آدرس و داده برای دسترسی به حافظه کد و حافظه داده جدا است.



16

معماری *Harvard*

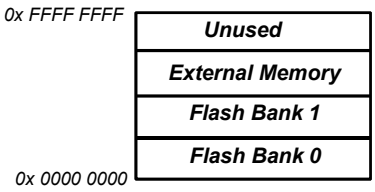
مزایا

پردازنده می تواند همزمان با اجرای دستوراتی که با حافظه داده در ارتباط هستند به واکشی دستور بعدی بپردازد

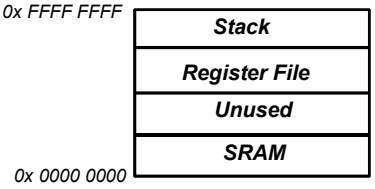
افزایش کارآیی و سرعت پردازش سیستم

17

تقسیم بندی فضای حافظه در معماری *Harvard*



Program Memory

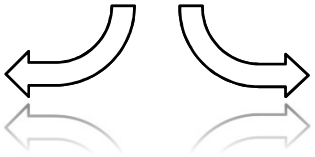


Data Memory

18

پردازنده های *RISC* و *CISC*

یکی از پارامترهای مهم در تقسیم بندی میکروپروسسورها نحوه کد کردن دستور العمل ها می باشد.



CISC
***C**omplex **I**nstruction **S**et*
***C**omputer*

RISC
***R**educed **I**nstruction **S**et*
***C**omputer*

19

ویژگی های پردازنده های *RISC*

- **تعداد دستورالعمل ها و پیچیدگی کد کردن آنها کاهش یافته، در مقابل سرعت اجرای آنها افزایش می یابد.**
- **به دلیل سادگی دستورات، اجرای آنها در یک سیکل کلاک انجام می شود.**
- **با توجه به سادگی دستورات، اجرای آنها توسط سخت افزار انجام می شود (کنترل سخت افزاری)**
- **بخش قابل توجهی از فضای تراشه به عنوان رجیسترها و حافظه نهان استفاده می شود.**
- **نوشتن برنامه اسمبلی آن مشکل است.**
- **استفاده ازروش *Pipeline* بسیار رایج است.**

20

ویژگی های پردازنده های *CISC*

- به دلیل تعداد زیاد دستورالعمل ها و پیچیدگی کد کردن دستورالعمل ها، سرعت اجرای آنها کاهش می یابد.
- به دلیل پیچیدگی دستورالعمل ها اجرای آنها در چندین سیکل کلاک انجام می شود.
- به دلیل پیچیدگی دستورها، اجرای آنها توسط میکروکد انجام می شود (کنترل ریزبرنامه نویسی).
- تعداد رجیسترها و نیز حافظه نهان، محدود می باشد.
- نوشتن برنامه اسمبلی آن ساده تر است.
- به ندرت از روش *Pipeline* برای آنها استفاده می شود.

21

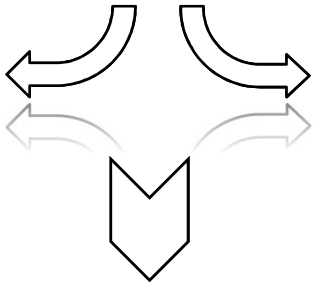
پردازنده های *DSP*

- معماری پردازنده های *DSP* برای انجام پردازش های دیجیتال مانند محاسبات *FFT*، فیلترهای دیجیتال و ... بهینه سازی شده اند.
- در این نوع پردازنده ها از واحد های ویژه *MAC* (*Multiplier Accumulator*) برای تسریع در انجام عملیات ضرب و جمع بر روی آرایه ای از اطلاعات نمونه برداری شده و نیز افزایش سرعت پردازش استفاده می شود.
- زمان لازم برای اجرای الگوریتم در تراشه *TMS320C6203* با فرکانس کلاک برابر با *MHz* ۳۰۰ بسیار کمتر از پردازنده *Pentium* با فرکانس کلاک برابر با *GHz* ۱ است.

22

پردازنده های *DSP*

در سیستم های تعبیه شده از پردازنده های *DSP* استفاده نمی شود



معمولاً فاقد امکانات جانبی مانند *ADC*، *DAC* و پورت های *USB* و *LAN* می باشند.

قیمت آنها بسیار بیشتر از میکروکنترلرها می باشد

در سیستم های پردازش گر صوت و تصویر استفاده می شوند

23

میکرو کنترلرها

- **با افزودن حافظه و ادوات I/O به میکروپروسسورها و قرار دادن آنها در یک تراشه، میکروکنترلرها به وجود می آیند.**
- **عملکرد آنها مشابه با کامپیوترهای شخصی و ادوات جانبی در مقیاس کوچکتر می باشد.**
- **به دلیل برخورداری از کلیه بخش های مورد نیاز، استفاده از آنها در سیستم های تعبیه شده بسیار رایج است.**

24

میکرو کنترلرها

- بلوک هایی که معمولاً در میکروکنترلرها مورد استفاده قرار می گیرند:
  - مبدل ADC
  - پورت سریال
  - تایمر (Timer)
  - مولد PWM
  - پورت های موازی
  - کنترل کننده وقفه
  - Watch dog Timer

25

انواع میکروکنترلرها با توجه به حوزه کاربرد

- میکروکنترلرهای ۴ بیتی
  - از حافظه های چند صد بایتی استفاده می کنند
  - معماری ساده ای دارند
  - مصرف توان آنها کم است
  - کاربرد آنها در سیستم های از قبیل کنترل کننده های راه دور، سیستم های هشداردهنده، کنترلر صفحه کلید کامپیوتر و موارد مشابه می باشد.
  - نمونه ای از این تراشه ها را می توان میکروکنترلرهای سری MARC4 محصول شرکت ATmel را نام برد.

26

انواع میکروکنترلرها با توجه به حوزه کاربرد

- میکروکنترلرهای ۸ بیتی
  - از رایج ترین انواع پردازنده می باشند.
  - حافظه به کار رفته در آنها در حد چند کیلوبایت است.
  - برنامه نویسی این تراشه ها توسط زبان اسمبلی و یا زبان های سطح بالا از قبیل Basic و C انجام می شود.
  - نسل قدیمی این میکروکنترلرها از قبیل 86HC11 و 8051 از معماری CISC استفاده می کردند؛ اما تراشه های جدید از قبیل خانواده PIC و AVR از معماری RISC استفاده می کنند.

27

انواع میکروکنترلرها با توجه به حوزه کاربرد

- میکروکنترلرهای ۱۶ و ۳۲ بیتی
  - حجم حافظه به کار رفته در آنها از چند ده کیلو بایت تا چند صد مگابایت می باشد.
  - به کار گیری آنها در سیستم های تعبیه شده بسیار رایج است.
  - برنامه نویسی آنها معمولاً توسط زبان C انجام می شود.
  - در برخی موارد، سیستم عامل های بلادرنگ مانند Linux و Windows CE به عنوان هسته نرم افزاری جهت مدیریت برنامه کاربر در این تراشه ها مورد استفاده قرار می گیرند.
  - از معروف ترین میکروکنترلرهای ۳۲ بیتی می توان به پردازنده ARM اشاره نمود

28

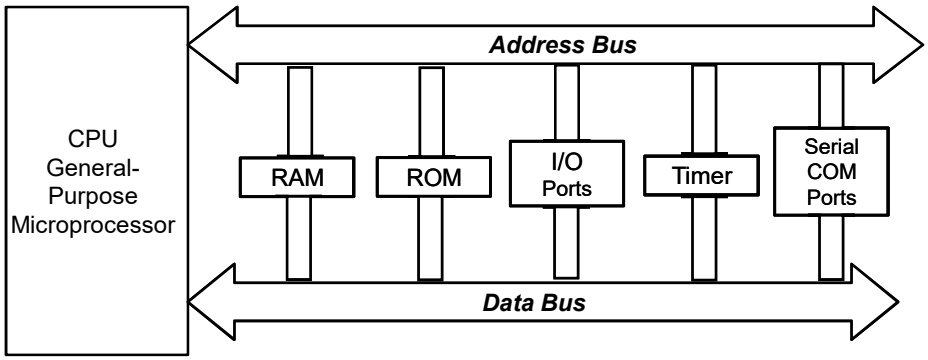
میکروکنترلرها در برابر میکروپروسسورهای همه منظوره

- منظور از میکروپروسسور، میکروپروسسورهای همه منظوره مانند X86 شرکت Intel (8086، 80386، 80486، Pentium) یا خانواده PowerPC شرکت موتورولا است..
- ایم میکروپروسسورها هیچ گونه RAM و یا ROM و یا پورت های I/O داخلی بر روی تراشه خود ندارند؛ به همین دلیل معمولاً آنها را میکروپروسسورهای همه منظوره می نامند.
- یک میکروکنترلر از یک CPU (میکروپروسسور) به علاوه مقدار ثابتی RAM، ROM، پورت های I/O و تایمر تشکیل شده است و در نتیجه، طراح نمی تواند آنها را به صورت خارجی به پروسسور اضافه کند.

29

میکروکنترلرها در برابر میکروپروسسورهای همه منظوره

- سیستم یک میکروپروسسور همه منظوره



30

میکروکنترلرها در برابر میکروپروسسورهای همه منظوره

- سیستم یک میکروکنترلر

CPU	RAM	ROM
I/O and ADC	Timer	Serial COM Port

31

میکروکنترلر AVR

- معماری اولیه AVR توسط دو دانشجوی مؤسسه فن آوری نروژ (NHT) به نام های Alf Egil Bogen و Vegard Wallan طراحی و سپس در سال ۱۹۹۶ توسط شرکت اینتل خریداری و تولید شد.

- معنای کلمه AVR بر کسی معلوم نیست. شرکت Atmel می گوید که کلمه AVR چیزی غیر از نام یک محصول نیست.
- البته برخی بر این باور هستند که کلمه AVR مخفف عبارت **A**dvanced **V**irtual **R**ISC یا **A**lf, **V**egard, **R**ISC است.

32

میکروکنترلر AVR

- انواع مختلفی از AVR با ویژگی های متفاوت وجود دارد.
- به جز AVR32 که یک میکروکنترلر ۳۲ بیتی است، بقیه AVRها همگی ۸ بیتی هستند یعنی CPU در هر لحظه فقط با ۸ بیت داده می تواند کار کند.
- داده های بزرگتر از ۸ بیت به بخش های ۸ بیتی تقسیم شده و به وسیله CPU پردازش می شوند.

33

مروری بر خانواده میکروکنترلر AVR

خانواده میکروکنترلر AVR

- خانواده AVR کلاسیک (AT90Sxxxx)
- خانواده Tiny AVR (ATtinyxxxx)
- خانواده Mega AVR (ATmegaxxxx)
- خانواده AVRهای تک منظوره

34

شماره تولید میکروکنترلر AVR

- تمام شماره های تولید با AT که دلالت بر Atmel دارند، شروع می شوند.
- اگر در عددهایی که در انتهای شماره تولید قرار دارند، از سمت چپ به راست بزرگترین عدد توان ۲ را پیدا کنیم، این عدد مقدار ROM را نشان می دهد.

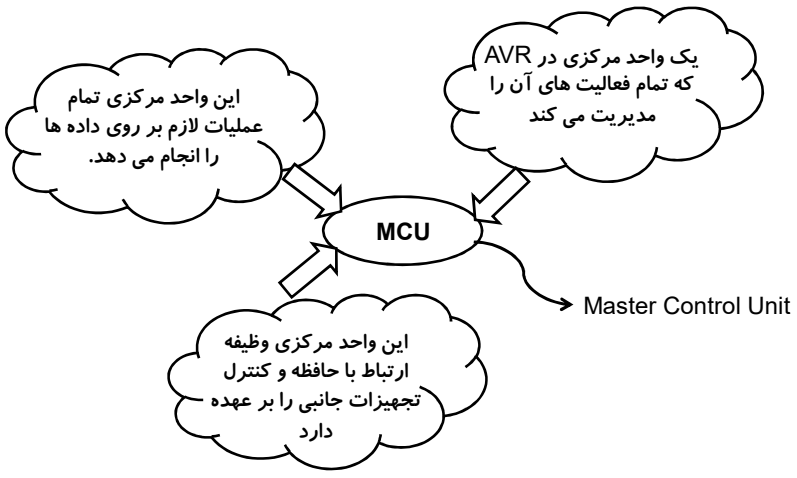
Atmega 128 → بزرگترین توان ۲ = 128 موجود → ROM مقدار = ۱۲۸ Kbytes

ATtiny 44 → بزرگترین توان ۲ = 4 موجود → ROM مقدار = ۴ Kbytes

AT90PWM216 → بزرگترین توان ۲ = 16 موجود → ROM مقدار = ۱۶ Kbytes

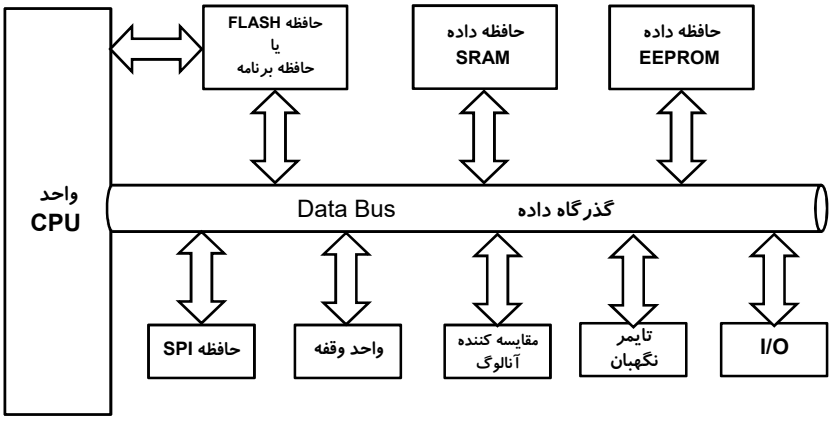
35

ساختار پردازنده میکروکنترلر AVR



36

واحد MCU



37

معماری میکروکنترلر AVR

- در میکروکنترلرهای AVR از معماری هاروارد استفاده شده است.
- این معماری در مقایسه با معماری سنتی Von-Neumann پهنای باند بهتری دارد.
- AVR یک میکروکنترلر تک تراشه ای با معماری ۸ بیتی RISC است.
- از امکانات استاندارد این میکروکنترلر می توان ROM برنامه (کد)، RAM داده، E^2PROM داده، تایمرها، پورت های I/O داخلی را نام برد.

38

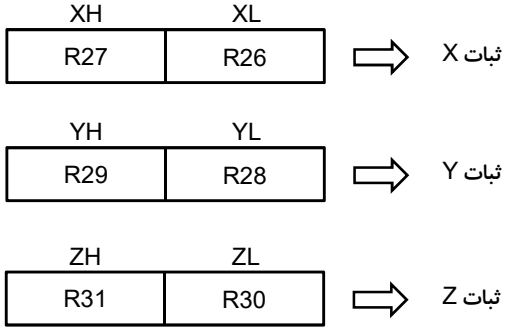
ثبات های عمومی (فایل رجیستر)

- CPU از ثبات های بسیاری برای ذخیره سازی موقت داده استفاده می کند.
- در AVR، ۳۲ ثبات همه منظوره (GPR) وجود دارد که R0 تا R31 نام دارند.
- این ثبات ها، پایین ترین مکان حافظه را به خود اختصاص داده اند.
- این ثبات ها، ۸ بیتی بوده و با واحد ALU در ارتباط مستقیم می باشند.

39

ثبات های عمومی (فایل رجیستر)

- کاربرد ثبات های X و Y و Z که به آنها اشاره گر پشته گفته می شود، علاوه بر کاربرد عمومی برای آدرس دهی غیرمستقیم در فضای حافظه داده و برنامه به کار می روند.



40

سازمان دهی حافظه در AVR

- حافظه های اصلی در معماری کلی AVR
  - فضای حافظه برنامه
  - فضای حافظه داده
    - SRAM داده
    - E^2PROM داخلی

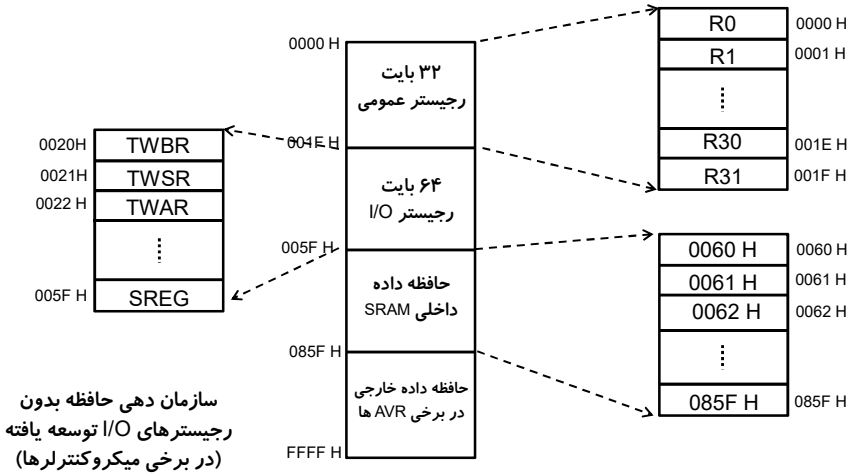
41

تقسیم بندی حافظه داده

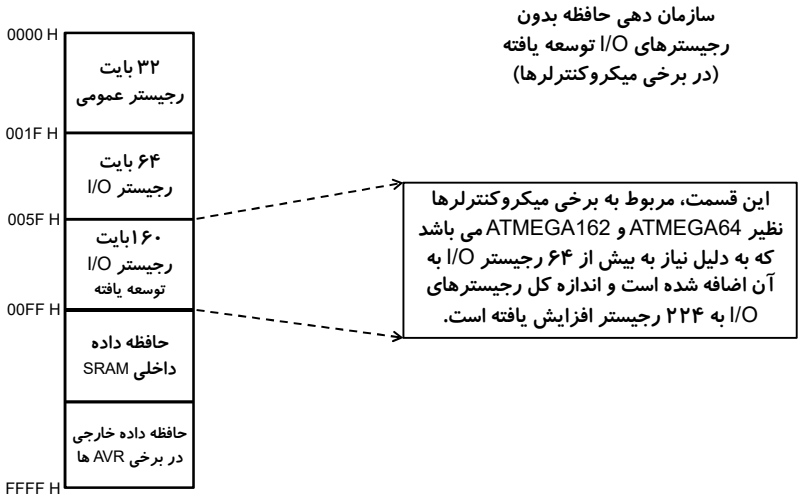
- حافظه داده
  - ۳۲ ثبات عمومی
  - ۶۴ ثبات I/O
  - حافظه داده داخلی SRAM
  - حافظه داده خارجی (برای برخی از میکروکنترلرها مانند ATmega 128)

42

حافظه داده



حافظه داده



ثبات های I/O

- ثبات های I/O از مهم ترین ثبات هایCPU بوده و برای کاربردهای خاصی مانند تایمرها، ارتباط های سریال، پورت های I/O، ADC و ... استفاده می شوند.
- تعداد مکان های حافظه برای این ثبات ها به تعداد پایه ها و کارکردهای جانبی که توسط تراشه پشتیبانی می شود، بستگی دارد.
- تعداد مکان های حافظه برای این ثبات ها از تراشه ای به تراشه دیگر حتی در بین اعضای یک خانواده می تواند تغییر کند، اما تمام AVR ها حداقل ۶۴ بایت فضای حافظه I/O دارند.
- به این ۶۴ بایت، حافظه استاندارد I/O گفته می شود.

45

ثبات های I/O

- برخی از آدرس های حافظه برای مصارف آینده رزرو شده اند.
- در AVR های با بیش از ۳۲ پایه مانند ATmega64، ATmega128 و ATmega256 یک حافظه I/O توسعه یافته وجود دارد (۱۶۰ بایت) که برای کنترل پورت های اضافه در وسایل جانبی اضافه می باشد مانند ثبات کنترلی مربوط به PortG در ATmega128.
- در میکروکنترلرهای سری ATmega2561، ATmega640، ATmega1280، ATmega1281 و ATmega2560 این بخش به ۴۱۶ بایت افزایش یافته است.
- در دیگر میکروکنترلرها، ثبات های I/O ثبات های SFR نامیده می شوند زیرا هریک هر یک به کارکرد ویژه ای اختصاص یافته اند.

46

حافظه داده داخلی SRAM

- فضای حافظه داده داخلی پس از ثبات های I/O شروع شده و با توجه به نوع میکروکنترلر متفاوت است.
- SRAM داخلی به طور گسترده ای برای ذخیره سازی داده و پارامترها به وسیله برنامه AVR و کامپایلرهای C به کار گرفته می شود. به همین دلیل آن را حافظه موقت (Scratch Pad) نیز می نامند.
- هر خانه SRAM به طور مستقیم از طریق آدرسش قابل دسترسی است.
- طول هر خانه این حافظه ۸ بیت است.
- اندازه SRAM نیز همانند حافظه I/O از تراشه ای به تراشه دیگر حتی در بین اعضای یک خانواده می تواند تغییر کند.

47

حافظه داده خارجی

- اگر به توسعه حافظه نیاز باشد، این امکان در برخی از میکروکنترلرهای AVR وجود دارد که علاوه بر حافظه داخلی از KB ۶۴ حافظه داده خارجی نیز استفاده گردد.

48

SRAM در مقابل E^2PROM در تراشه های AVR

- از حافظه E^2PROM برای ذخیره داده استفاده می شود و با قطع برق اطلاعات خود را از دست نمی دهد درحالی که SRAM چنین نیست.
- به همین دلیل از E^2PROM برای ذخیره داده هایی که به ندرت تغییر می کنند و نباید اطلاعاتشان با خاموش شدن سیستم از بین برود، استفاده می شود. مانند option ها و تنظیمات سیستم.

49

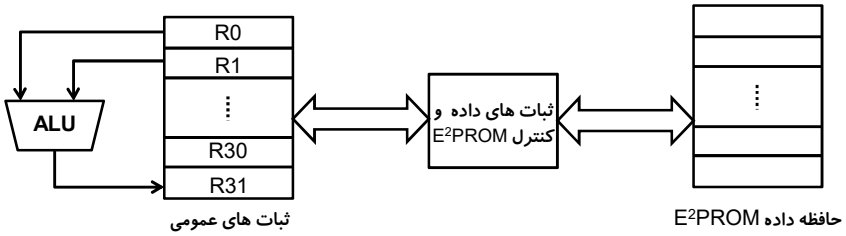
SRAM در مقابل E^2PROM در تراشه های AVR

- از آنجایی که از SRAM برای ذخیره داده‌ها و پارامترهایی استفاده می شود که بارها تغییر می کنند، سه بخش تشکیل دهنده حافظه داده (ثبات های عمومی، حافظه I/O، SRAM داخلی داده) از SRAM ساخته شده اند.
- در AVR منظور از E^2PROM مقدار حافظه E^2PROM و منظور از SRAM، مقدار حافظه RAM داخلی داده است.

50

SRAM در مقابل E^2PROM در تراشه های AVR

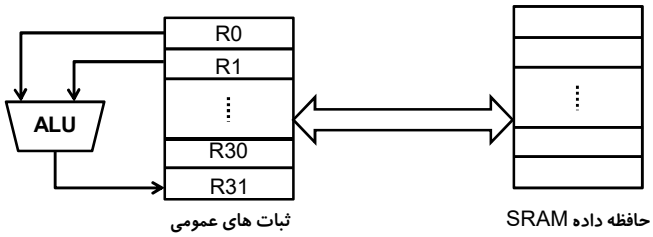
- واحد CPU به طور مستقیم به فضای حافظه داده E^2PROM دسترسی ندارد، بلکه توسط ثبات های واسط (R0 تا R31) و ثبات های کنترلی به آن دسترسی خواهد داشت.



51

SRAM در مقابل E^2PROM در تراشه های AVR

- حافظه SRAM به طور مستقیم در اختیار CPU نمی باشد و برای دسترسی به آن از یک ثبات واسط (R0 تا R31) استفاده می شود.
- استفاده از SRAM برای انجام عملیات، زمان بیشتری نسبت به حالتی که عملیات فقط روی ثبات ها عمومی انجام می شود نیاز دارد.



52

اندازه حافظه داده در برخی از تراشه های AVR

شماره قطعه	حافظه داده (بایت) =	ثبات های I/O (بایت) +	SRAM (بایت) +	ثبات های عمومی
Tiny25	۲۲۴	۶۴	۱۲۸	۳۲
Tiny85	۶۰۸	۶۴	۵۱۲	۳۲
ATmega8	۱۱۲۰	۶۴	۱۰۲۴	۳۲
ATmega16	۱۱۲۰	۶۴	۱۰۲۴	۳۲
ATmega32	۲۱۴۴	۶۴	۲۰۴۸	۳۲
ATmega128	۴۳۵۲	۱۶۰+۶۴	۴۰۹۶	۳۲
ATmega2560	۸۷۰۴	۴۱۶+۶۴	۸۱۹۲	۳۲

53

فضای حافظه برنامه

- در میکروکنترلرها از ROM برای ذخیره سازی برنامه استفاده می شود؛ به همین دلیل ROM برنامه یا کد خوانده می شود.
- اگر چه AVR دارای ۸ MB فضای ROM برنامه یا کد است ولی این مقدار ROM روی تمام اعضای سری AVR نصب نمی شود.
- اندازه ROM برحسب نوع سری از ۱ KB تا ۲۵۶ KB تغییر می کند.

54

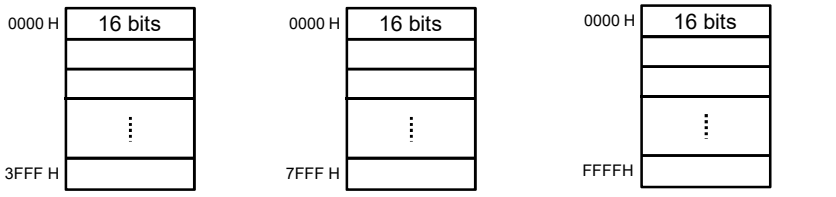
فضای حافظه برنامه

- میکروکنترلر AVR یکی از اولین خانواده میکروکنترلرها است که از حافظه سریع و آنی (Flash) برای ذخیره سازی برنامه استفاده می کند.
- این حافظه آنی برای تولید سریع محصولات مناسب است زیرا در طی چند ثانیه پاک می شود.

55

فضای حافظه برنامه

- از آنجایی که تمام دستورات AVR، ۱۶ یا ۳۲ بیت عرض دارند، حافظه برنامه به صورت زیر سازمان دهی می شود:



56

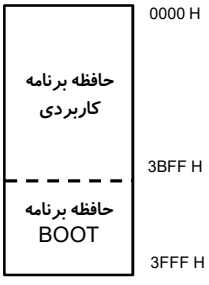
تقسیم بندی فضای حافظه برنامه

- فضای حافظه برنامه flash
  - بخش برنامه کاربردی
  - بخش BOOT

57

تقسیم بندی فضای حافظه برنامه

- فضای داخلی حافظه flash در ATmega32



با استفاده از فیوز بیت های BOOTSZ1 و BOOTSZ2 می توان اندازه حافظه BOOT را مشخص نمود.

58

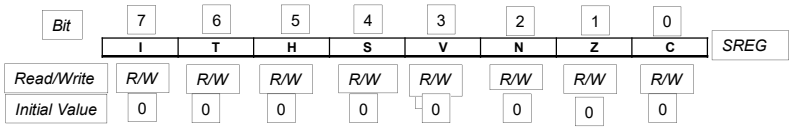
فضای حافظه برنامه

- **حافظه برنامه flash قابلیت تا ۱۰۰۰۰ مرتبه نوشتن (پاک کردن) را دارد.**
- **در خانواده AVR، همه اعضا هنگام روشن شدن، بدون توجه به دسته بندی ها و مدل های مختلف از خانه 0000$ شروع به کار می کنند.**
- **یعنی انتظار می رود که اولین کد عمل (Opcode) در آدرس 0000H حافظه ROM ذخیره شده باشد.**
- **برای قرار دادن اولین کد عمل در خانه 0000H حافظه ROM از دستورالعمل ORG استفاده می شود.**

59

ثبات وضعیت

- **ثبات وضعیت (SREG) یک ثبات ۸ بیتی است که به ثبات پرچم نیز معروف است.**
- **بیت های این ثبات تحت تأثیر برخی عملیات CPU (ریاضی و منطقی) فعال می شوند بدین معنا که برخی از شرایطی که پس از اجرای یک دستور به وجود می آیند را نشان می دهند.**



60

ثبات وضعیت

I	T	H	S	V	N	Z	C	SREG

- **پرچم نقلی (C): این پرچم هنگام خارج شدن رقم نقلی از بیت هفتم، یک می شود. این پرچم بین انجام عمل محاسباتی و منطقی تأثیر می پذیرد.**
- **پرچم صفر (Z): این پرچم، هنگام صفر شدن عملیات محاسباتی یا منطقی، یک خواهد شد.**
- **پرچم منفی (N): این پرچم، هنگام منفی شدن عملیات محاسباتی یا منطقی، یک خواهد شد.**
- **پرچم سرریز (V): این پرچم هنگامی که نتیجه عملیات علامت دار، بزرگتر از ۸ باشد، یک می شود. یک شدن این بیت بیانگر این است که نتیجه محاسباتی صحیح نیست.**

61

ثبات وضعیت

I	T	H	S	V	N	Z	C	SREG

- **پرچم علامت (S): این پرچم همواره نتیجه XOR بین پرچم های V و N می باشد.**
- **پرچم نیم نقلی (نقلی کمکی) (H): اگر هنگام عملیات جمع، یک انتقال از بیت ۳ به ۴ در یکی از ثبات های عمومی یا نتیجه نیم بایت پایین (بیت های ۰ تا ۳) بزرگتر از ۹ باشد، پرچم H یک خواهد شد و برای محاسبه به صورت BCD مفید است.**
- **پرچم موقت (T): هنگام استفاده از دستورهای اسمبلی کپی کردن بیت ها، از این بیت به عنوان مبدأ یا مقصد استفاده می شود.**
- **پرچم فعال ساز وقفه سراسری (I): در صورت یک کردن این بیت، وقفه سراسری فعال می شود. اگر این بیت صفر باشد، هیچ وقفه ای رخ نمی دهد.**

62

پورت I/O

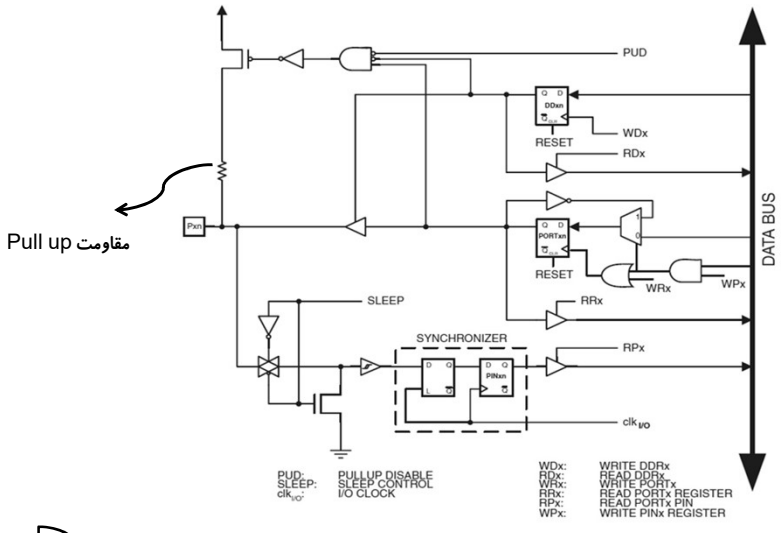
- پورت ها امکان دسترسی میکروکنترلر با دنیای خارج را فراهم می کنند.
- یک میکروکنترلر به کمک پورت های I/O می تواند داده های مختلفی را از دنیای خارج دریافت یا به دنیای خارج ارسال نماید.
- در میکروکنترلرهای AVR با توجه به نوع میکروکنترلر، تعداد پورت ها متفاوت می باشد.

نوع میکرو	ATTiny	ATmega128	ATmega256
تعداد پورت I/O	۶	۵۳	۸۶

- در AVR ها پورت ها به صورت PORTA، PORTB، PORTC و ... نام گذاری می شوند که هر کدام به صورت ۸ بیتی عمل می کنند.

63

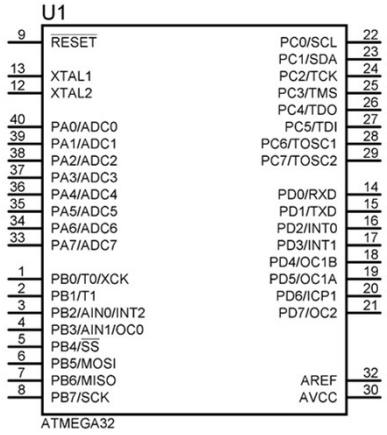
ساختار پورت I/O



64

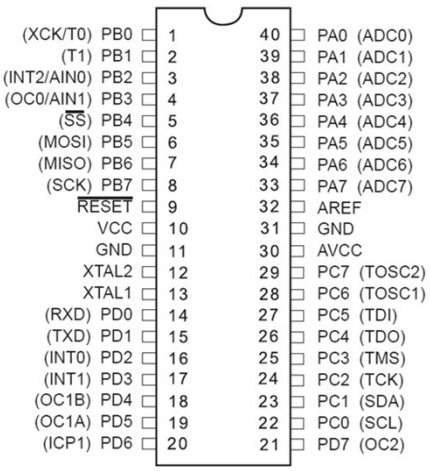
ساختار پورت I/O

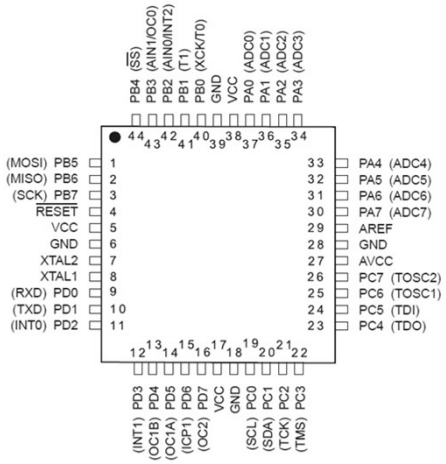
- پورت ها علاوه بر کاربرد عمومی به عنوان I/O معمولاً دارای وظایف خاصی نیز هستند که با توجه به AVR مورد نظر متفاوت است.



ساختار پورت I/O

Atmega32

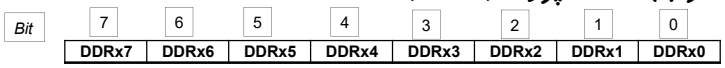




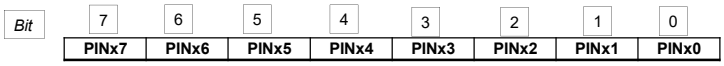
ساختار پورت I/O

- رجیستر داده پورت (PORTx)

- رجیستر جهت داده پورت (DDRx)



- رجیستر پایه های داده پورت (PINx)



67

ساختار پورت I/O

در عبارت های PORTx.y، DDRx.y **و** PINx.y **متغیر** x **بسته به نوع میکرو می تواند نام یکی از پورت های** A، B، C، D، E، F **و** G **باشد و** y **شماره یکی از پایه های صفر تا ۷ می باشد.**

68

ساختار پورت I/O

- تعیین جهت پایه ها به صورت ورودی و خروجی
  - برای این کار از رجیستر DDRx استفاده می شود.
  - اگر در بیت مورد نظر از این رجیستر صفر منطقی نوشته شودآن پایه به صورت ورودی تعریف می شود.
  - اگر در بیت مورد نظر از این رجیستر یک منطقی نوشته شودآن پایه به صورت خروجی تعریف می شود.

69

ساختار پورت I/O

- تعیین جهت پایه ها به صورت ورودی و خروجی
- مثال :

DDRA = 47 H	0	1	0	0	0	1	1	1

در این مثال، پایه های ۰ و ۱ و ۲ و ۶ از پورت A به صورت خروجی و پایه های ۳ و ۴ و ۵ و ۷ به صورت ورودی تعریف می شود.

70

ساختار پورت I/O

- **مقاومت بالاکش (*Pull-Up*)**
- **در مدار داخلی یک پین از میکروکنترلر، هر یک از پین های I/O دارای دو دیود محافظ برای محافظت از ولتاژ ورودی منفی یا بزرگتر از VCC است و همچنین دارای یک مقاومت بالاکش داخلی می باشد. مقاومت های بالاکش را می توان به دلخواه فعال یا غیر فعال کرد.**

- **هنگامی که پایه ای به صورت ورودی تعریف می شود (DDRx.n =0)، اگر در بیت PORTx.n مقدار یک منطقی و بیت PUD در رجیستر SFIOR نیز صفر منطقی نوشته شود، مقاومت بالاکش داخلی فعال خواهد بود.**

71

ساختار پورت I/O (مقاومت بالاکش)

- **رجیستر SFIOR**

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **هنگامی که بیت PUD برابر یک می گردد، مقاومت بالاکش درون پورت های I/O غیرفعال می شود، حتی اگر DDRx.n = 1 و PORTx.n = 1 شده باشد.**

72

ساختار پورت I/O (مقاومت بالاکش)

- وضعیت مختلف یک پایه در میکروکنترلر AVR

DDRx.n / PORTx.n	0	1
0	ورودی و امپدانس بالا	خروجی با مقدار صفر
1	ورودی با فعال بودن مقاومت بالاکش	خروجی با مقدار ۱

73

ساختار پورت I/O (مقاومت بالاکش)

- وضعیت مختلف یک پایه در میکروکنترلر AVR

DDRx.n	PORTx.n	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	x	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Px.n will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	x	Output	No	Output Low (Sink)
1	1	x	Output	No	Output High (Source)

74

ساختار پورت I/O

- هنگامی که پایه به صورت ورودی تعریف می شود، در صورتی که مقاومت بالاکش داخلی یا خارجی فعال نباشد، پایه I/O به حالت Hi-Z رفته و مقدار خوانده شده از ورودی پایه قابل اطمینان نمی باشد. بنابراین توصیه می شود هنگام ورودی کردن پایه، مقاومت بالاکش داخلی یا خارجی برای آن در نظر گرفته شود.
- هنگامی که پایه به صورت خروجی تعریف شده باشد، مقاومت بالاکش غیرفعال خواهد شد.

75

ساختار پورت I/O (مقاومت بالاکش)

- مثال

DDRB = 47 H	0	1	0	0	0	1	1	1
PORTB = C9 H	1	1	0	0	1	0	0	1

- در این مثال، پایه های ۰ و ۱ و ۲ و ۶ از پورت B به صورت خروجی و بقیه پایه مطابق جدول زیر به صورت ورودی تعریف می شود :

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
ورودی با فعال بودن Pull-up	خروجی با مقدار ۱	ورودی با غیرفعال بودن Pull-up	ورودی با غیرفعال بودن Pull-up	ورودی با فعال بودن Pull-up	خروجی با مقدار صفر	خروجی با مقدار صفر	خروجی با مقدار ۱

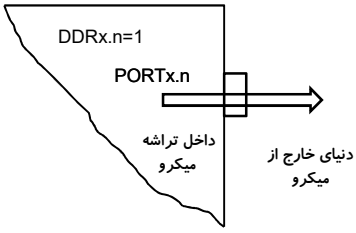
76

ساختار پورت I/O

- **نوشتن در پایه ها**

– ابتدا جهت پایه مورد نظر را به صورت خروجی تعریف می کنیم (DDRx.n=1)

– با نوشتن در بیت PORTx.n مقدار دلخواه را بر روی پایه موردنظر می نویسیم.

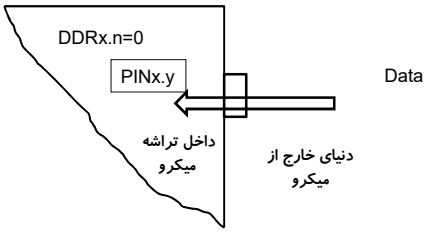


77

ساختار پورت I/O

- **خواندن از پورت**

– برای این کار می توان به کمک PINx اطلاعات را از روی پایه دریافت نمود.



78

سیستم توزیع پالس ساعت در AVR

- پالس ساعت داخلی از منابع مختلفی ایجاد می شود.
- این منابع می تواند نوسان ساز کریستالی، نوسان ساز RC خارجی، نوسان ساز کریستالی فرکانس پایین، نوسان ساز RC داخلی و پالس ساعت خارجی را نام برد.
- پالس های ساعت تولید شده از منابع، توسط یک Clock Multiplexer انتخاب می شود.
- پالس ساعت منبع انتخاب شده به قسمت AVR Clock Control Unit وارد می شود. این قسمت، وظیفه پخش پالس ساعت به قسمت های مختلف AVR را بر عهده دارد.

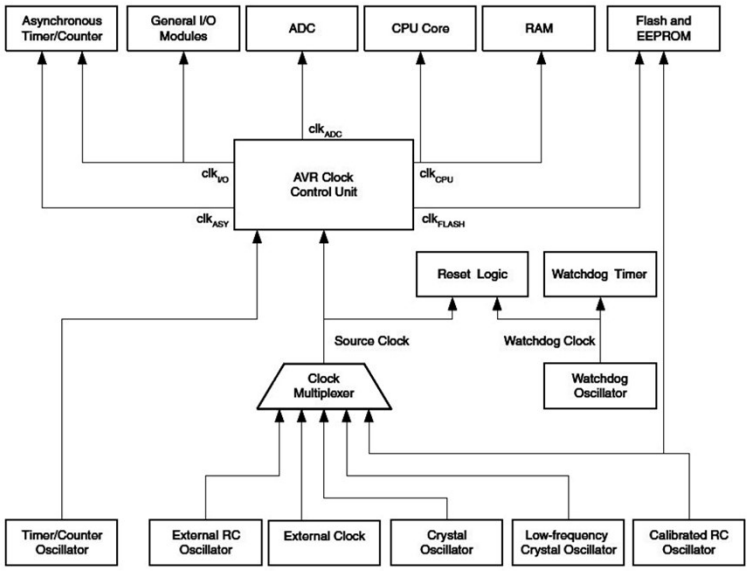
79

سیستم توزیع پالس ساعت در AVR

- دسته بندی پالس های ساعت ایجاد شده توسط واحد Clock Control :
  - پالس ساعت CPU (CLKCPU) : این پالس ساعت به بخش های اصلی AVR مانند CPU، رجیسترهای عمومی، رجیستر وضعیت SREG و حافظه داده SRAM اعمال می شود.
  - پالس ساعت I/O (CLK I/O) : این پالس ساعت برای بخش هایی چون فضای حافظه I/O که برای کنترل قسمت هایی نظیر USART، SPI، تایمر شمارنده و وقفه های خارجی به کار می رود.
  - پالس ساعت آسنکرون (CLKASY) : این پالس ساعت برای راه اندازی آسنکرون تایمر/شمارنده (۲) توسط کریستال ساعت ۳۲/۷۶۸ KHz به کار می رود.
  - پالس ساعت حافظه FLASH (CLKFLASH) : این پالس ساعت برای کنترل ارتباط باحافظه FLASH به کار می رود و معمولاً همراه با CLKCPU فعال می شود.
  - پالس ساعت ADC (CLKADC) : این پالس ساعت برای مبدل آنالوگ به دیجیتال بوده و به صورت جداگانه تأمین می شود و امکان توقف پالس ساعت CPU و I/O را به منظور کاهش تأثیر نویز مدارات دیجیتال روی ADC و در نتیجه افزایش دقت آن فراهم می کند.

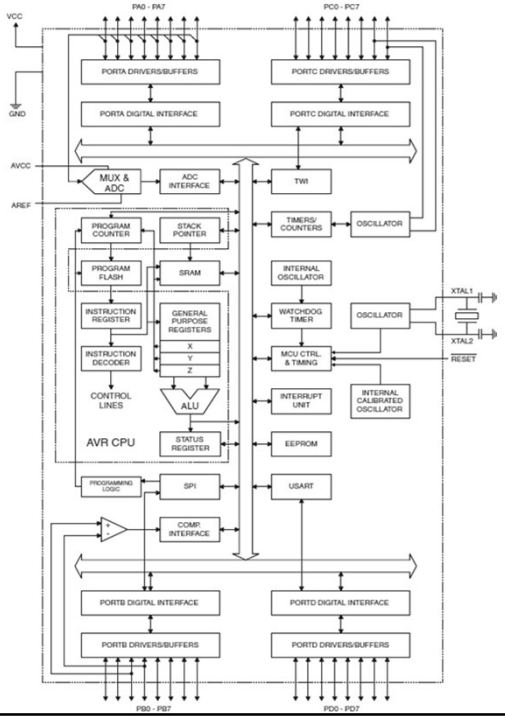
80

سیستم توزیع پالس ساعت در AVR



81

دیاگرام بلوکی AVR

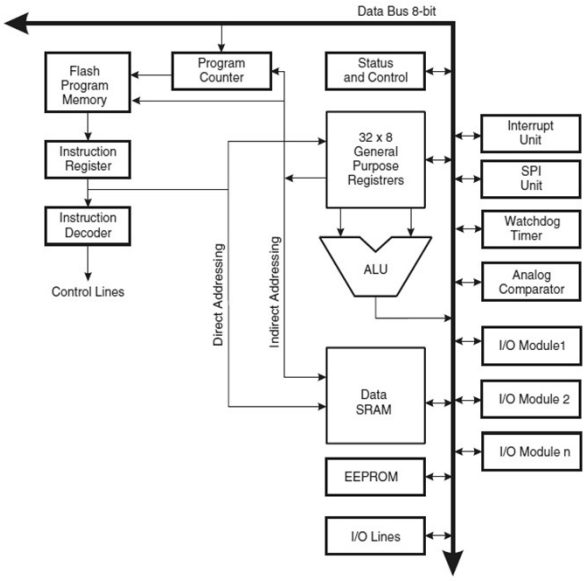


Atmega32

82

دیاگرام بلوکی واحد MCU

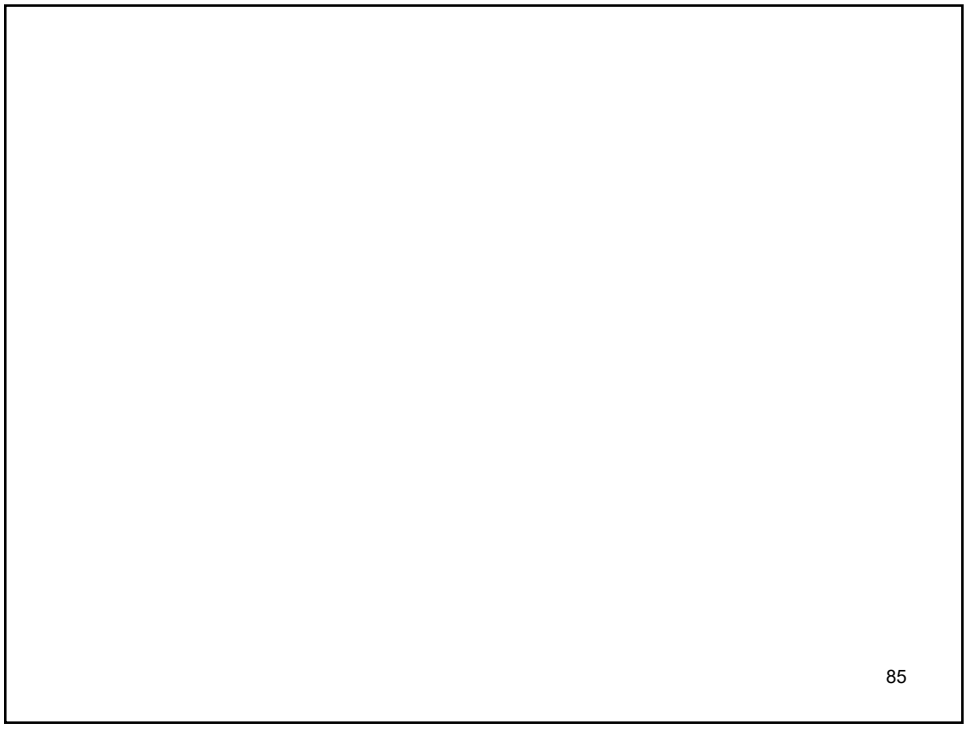
Atmega32



83

پایان فصل سوم

84


85