بنام خداوند جان و خرد

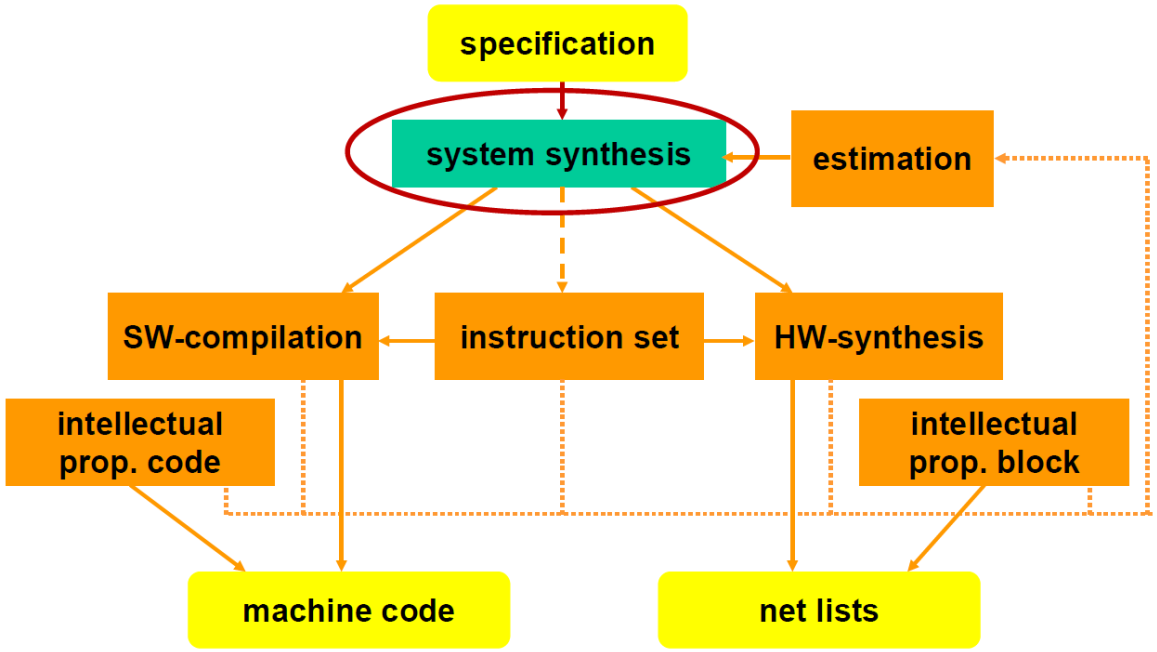
هم‌طراحی سخت‌افزار نرم‌افزار

جلسه پانزدهم: الگوریتم‌های ۲-Partitioning

ارائه‌دهنده: آتنا عبدی
a_abdi@kntu.ac.ir

مباحث این جلسه

- سنتز توام در روال هم‌طراحی سخت‌افزار و نرم‌افزار
  - الگوریتم‌های مکاشفه‌ای پایه
    - Cosyma
    - Global critical/Local phase



Optimization Strategy -روش‌های بخش‌بندی

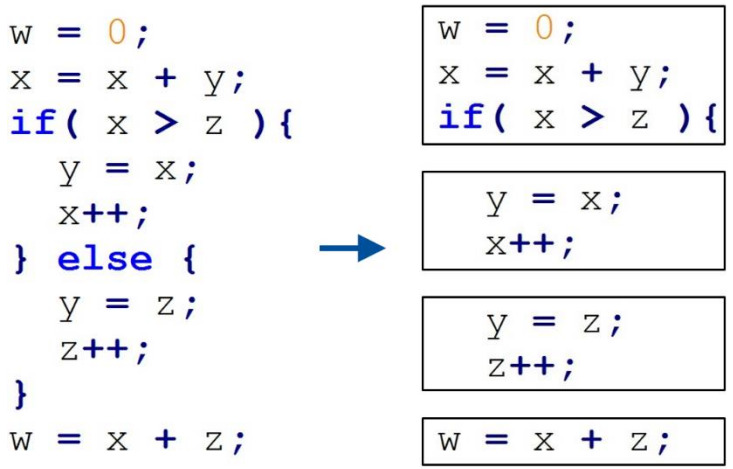
- دسته‌بندی روش‌ها براساس استراتژی که در رعایت محدودیت‌ها (کارایی-هزینه) دارند:
  - هدف اولیه: کارایی
  - Primal Strategy
  - **سیستم Vulcan:** تخصیص همه وظایف به ASIC و انتقال تدریجی توابع غیربحرانی به پردازنده با هدف کاهش هزینه
  - هدف اولیه: هزینه
  - Dual Strategy
  - **سیستم Cosyma:** تخصیص همه وظایف به پردازنده و انتقال تدریجی توابع بحرانی به سمت ASIC با هدف افزایش کارایی

Cosyma

- توسط Rolf Ernst در دانشگاه Braunschweig آلمان ارائه شد
  - شروع بخش‌بندی از یک راهکار مبتنی بر هزینه تماما نرم‌افزاری
  - حرکت تکراری بلوک‌ها به سمت واحدهای پردازشی سخت‌افزاری با هدف تحقق هدف کارایی
    - اجزای بحرانی را سخت‌افزار منتقل می‌کند
  - زبان توصیف سطح سیستم در این ابزار: C^x
    - نسخه توسعه یافته زبان C که قابلیت توصیف محدودیت زمانی، ارتباطات، پروسه‌ها و جریان سنتز توأم را دارد
    - نتیجه کامپایل ترکیبی از گراف جریان و گراف کنترل جریان است
      - مشابه CDFG

الگوریتم Cosyma

- واحد بخش‌بندی براساس گراف ساخته شده بلوک پایه (Basic Block) است
  - بلوک پایه: بلوک‌هایی از برنامه که در آن‌ها دستورات پرش وجود ندارد (branch-free blocks)
  - سطح ریزدانه‌گی الگوریتم، بلوک پایه است و این بخش‌ها بین سخت‌افزار و نرم‌افزار توزیع می‌شوند
- معماری هدف:
  - مجموعه‌ای از عناصر پردازشی متشکل از CPU و تجهیزات ASIC



الگوریتم Cosyma

- انتقال دادن بلوک‌های پایه به ASIC با هدف بهبود کارایی

- تغییر زمان اجرا (بهبود کارایی) ناشی از انتقال بلوک پایه مشخص از CPU به ASIC:

$$\Delta c(b) = w(t_{HW}(b) - t_{SW}(b) + t_{com}(Z) - t_{com}(Z \cup b)) \times It(b)$$

- $\Delta c(b)$: کاهش تقریبی زمان اجرا
- w: وزن
- $t_{HW}(b)$: زمان اجرا روی HW
- $t_{SW}(b)$: زمان اجرا روی SW
- $t_{com}(Z)$: زمان تقریبی ارتباط بین SW و HW
- $It(b)$: کل تعداد تکرار اجرای بلوک پایه

الگوریتم Cosyma

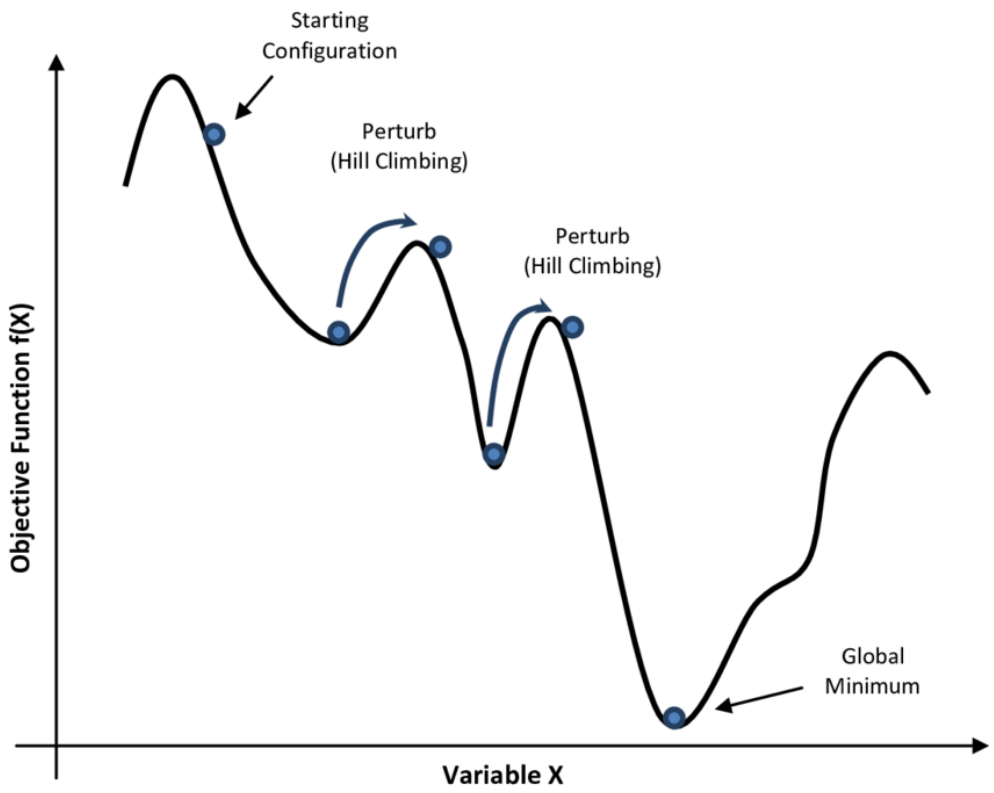
- چگونگی انتخاب بلوک‌های پایه برای انتقال به پردازش سخت‌افزاری:
  - هدف اصلی: تحقق محدودیت کارایی و کمینه کردن هزینه
  - مسئله جستجوی فضای طراحی با پیچیدگی NP-hard
  - حل این مسئله در الگوریتم Cosyma با روش Simulated Annealing
  - روش آماری برای جستجوی موثر فضای طراحی و بهینه‌سازی

الگوریتم Cosyma (ادامه)

- مراحل Simulated Annealing:
  - راهکار اولیه به‌صورت تصادفی تولید می‌شود و هزینه آن محاسبه می‌شود
  - راهکار تصادفی دیگری در همسایگی راهکار قبلی تولید می‌شود و هزینه آن نیز محاسبه می‌گردد
  - مقایسه هزینه این دو راهکار
    - اگر راهکار جدید بهتر بود، جایگزین وضعیت فعلی سیستم می‌شود
    - اگر راهکار جدید بهتر نبود، با احتمال مشخصی جایگزین وضعیت فعلی سیستم می‌شود
  - مجددا راهکاری در همسایگی انتخاب شده و این عملیات تکرار می‌شود تا به جواب دلخواه برسیم

الگوریتم Cosyma (ادامه)

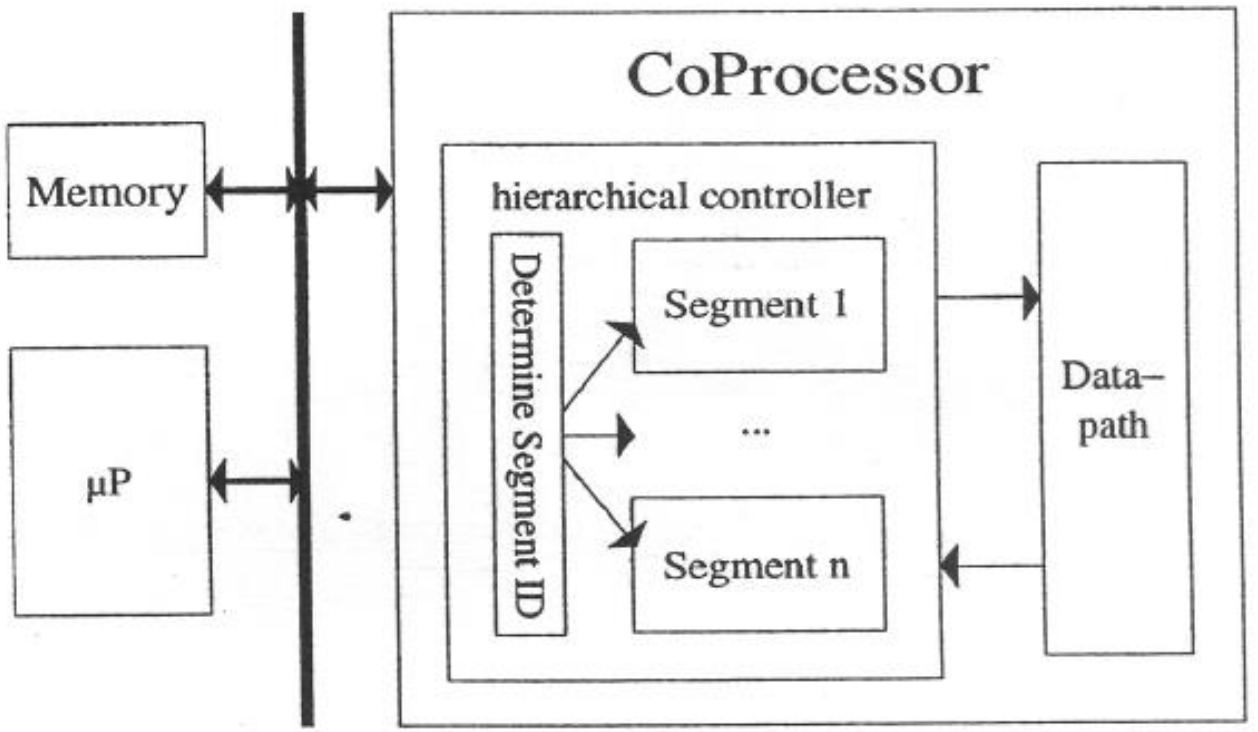
- مراحل Simulated Annealing:
  - راهکار: یک بخش‌بندی
  - تابع هزینه: بهبود زمان پایان اجرای سیستم
  - حرکت‌ها: انتقال BB به بخش دیگر
  - تابع احتمالی قبولی نمایی متناسب با سیستم



پیاده‌سازی الگوریتم Cosyma (ادامه)

- پس از مشخص شدن بلوک‌های سخت‌افزاری و نرم‌افزاری
  - بلوک‌های سخت‌افزاری در عناصر پردازشی ASIC قرار داده می‌شود
  - بلوک‌های نرم‌افزاری به زبان C برده می‌شوند
  - شماره سگمنت سخت‌افزاری بلوک‌های سخت‌افزاری در کد قرار داده می‌شود
  - آدرس خاصی به هر واحد پردازشی سخت‌افزاری داده می‌شود تا بتوان به آن مراجعه کرد
- با اجرای کد، بلوک‌های سخت‌افزاری و نرم‌افزاری اجرا می‌شوند

معماری سخت‌افزاری سیستم Cosyma



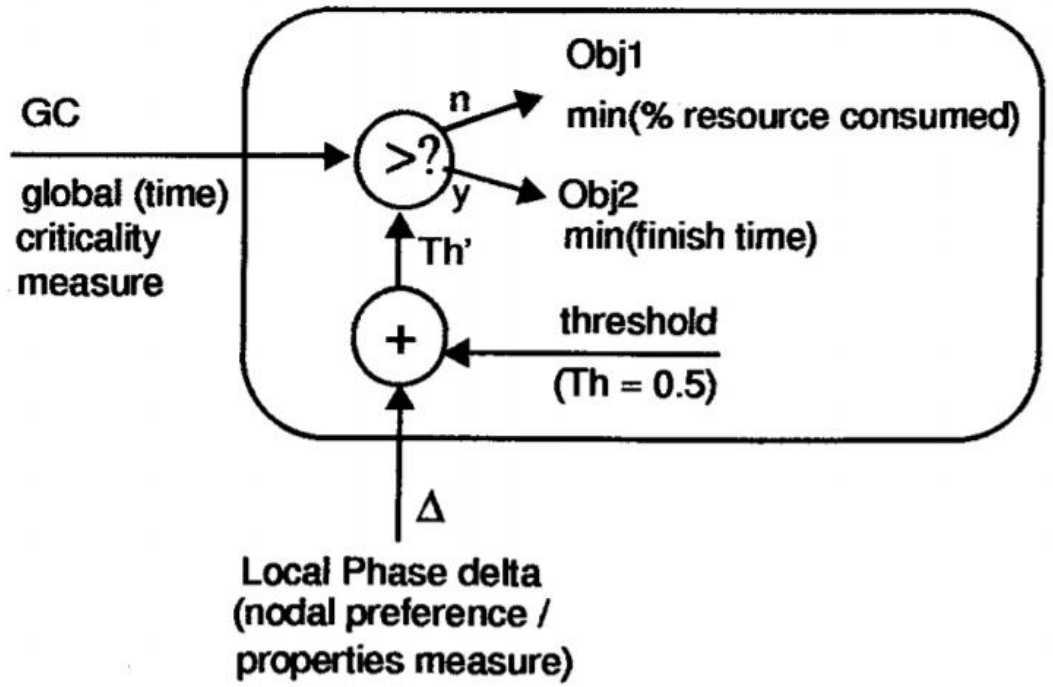
نتایج حاصل از الگوریتم Cosyma

- نتایج حاصل از اعمال الگوریتم Cosyma
  - بهبود سرعت ۲/۷ تا ۹/۷ برابر نسبت به پیاده‌سازی تمام نرم‌افزاری
  - در این نتایج، از الگوریتم‌های خاصی جهت افزایش اجرای موازی بلوک‌ها استفاده شده است
    - روش‌های مبتنی بر خط لوله در حلقه و محاسبات، روش‌های branch prediction
  - بدون اعمال روش‌های بهبود و افزایش موازی‌سازی،
    - سرعت بدست آمده حدود ۲ برابر بوده است

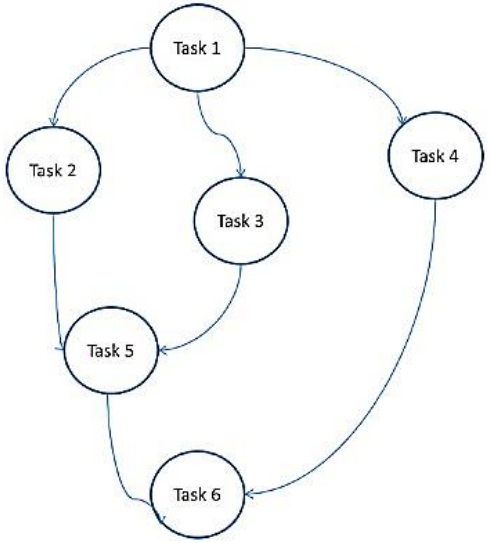
روش‌های بخش‌بندی دیگر

- روش Global critically/Local phase
  - رویکرد میانه‌تری در دو جنبه کارایی و هزینه دارد
  - معیار global critically: (هدف کارایی)
    - سنجش میزان بحرانی بودن گره‌های سیستم در زمان‌بندی جهت تخصیص به پردازشگر سخت‌افزاری
  - معیار local phase: (هدف هزینه)
    - مشخص کردن راهکار ارزان پیاده‌سازی یک تابع براساس شیوه پیاده‌سازی و صرف منابع کمتر
- با انتخاب صحیح معیارها، این روش پاسخ نزدیک به بهینه را در زمان مناسب تولید می‌کند

Global critically/Local phase روش

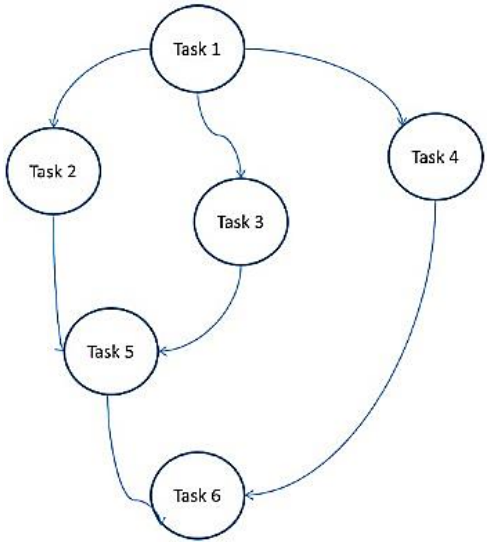


مثال

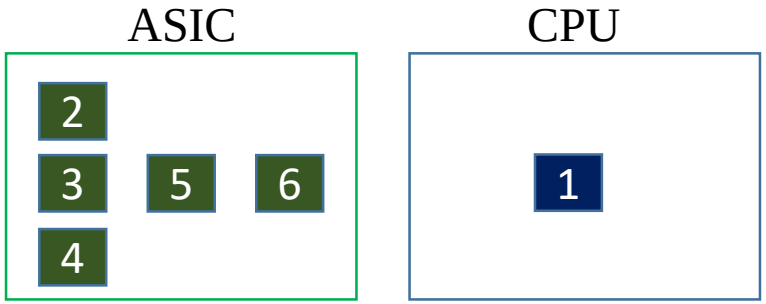


- گراف وظایف سیستم نمونه:
  - فرضیات: همه وظایف یکسان
    - زمان اجرا روی پردازنده ۱۰ و روی سخت‌افزار ۵
    - هزینه مساحت هر واحد سخت‌افزاری برابر x
    - موعد اجرا برابر ۳۵ و محدودیت مساحت سیستم برابر 3x
  - بخش‌بندی با روش‌های
    - Primal مانند Vulcan، Dual مانند Cosyma، ترکیبی مانند GC/LP

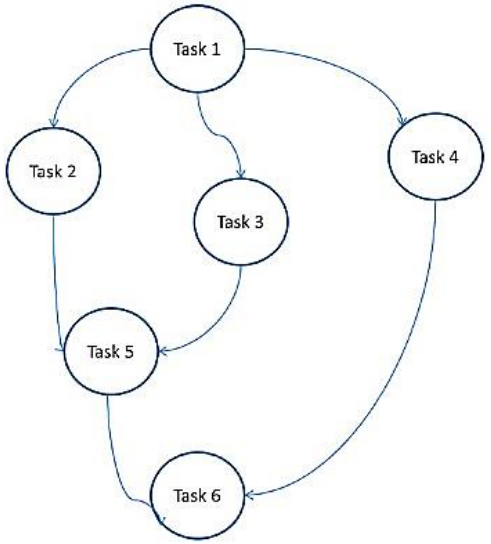
مثال (ادامه)



- روش Primal
  - مرحله ۱: تخصیص همه وظایف به سخت‌افزار
    - هزینه سخت‌افزاری: 6x
    - تاخیر اجرا: ۲۰ واحد به‌دلیل وابستگی داده‌ای
    - ...
  - مرحله نهایی: تخصیص وظیفه ۱ به پردازنده و بقیه به سخت‌افزار با لحاظ کردن وابستگی داده
    - هزینه سخت‌افزاری: 3x
    - تاخیر اجرا: ۲۵



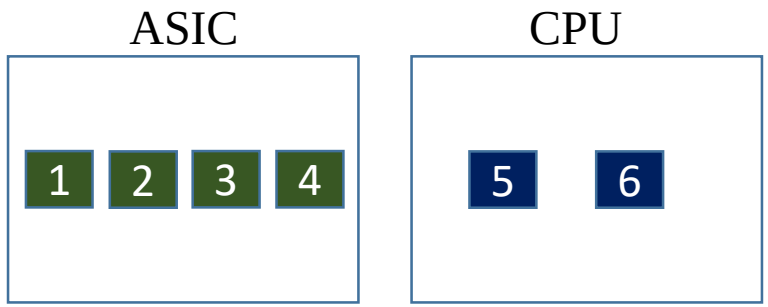
مثال (ادامه)



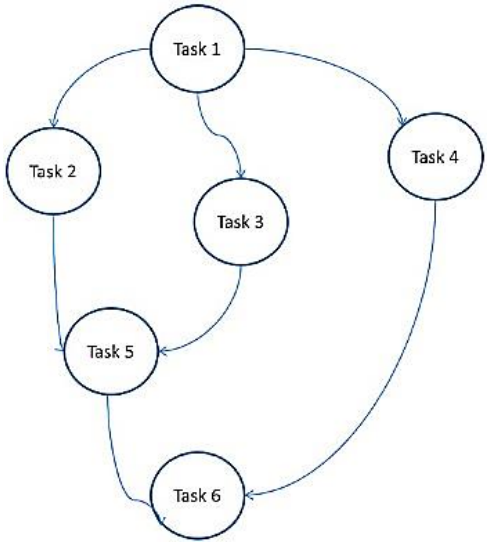
- روش Dual
  - مرحله ۱: تخصیص همه وظایف به نرم‌افزار
    - تاخیر اجرا: ۶۰ واحد
  - مرحله ۲: تخصیص وظیفه ۱ به سخت‌افزار
    - هزینه سخت‌افزاری: x
    - تاخیر اجرا: ۵۵

.....

  - مرحله آخر: تخصیص وظایف ۱، ۲، ۳ و ۴ به یک سخت‌افزار
    - هزینه سخت‌افزاری: ؟
    - تاخیر اجرا: ؟



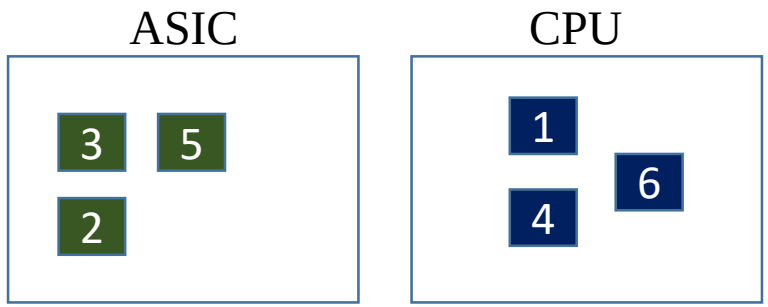
مثال (ادامه)



- روش ترکیبی
  - مرحله ۱: تخصیص وظیفه ۱ به نرم‌افزار به‌دلیل بحرانی نبودن
    - تاخیر اجرا: ۱۰ واحد
  - مرحله ۲: تخصیص وظیفه ۴ به نرم‌افزار به‌دلیل بحرانی نبودن
    - تاخیر اجرا: ۲۰
  - مرحله ۳: تخصیص وظیفه ۳ به سخت‌افزار به دلیل بحرانی بودن

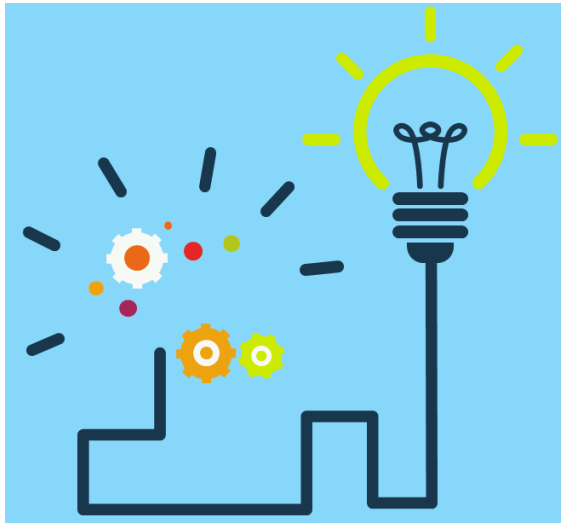
.....

  - مرحله آخر: افزودن وظایف ۲و ۵ به سخت‌افزار و ۶ به سخت‌افزار
    - هزینه سخت‌افزاری: 2x
    - تاخیر اجرا: ؟



مباحثی که این جلسه آموختیم

- فرایند سنتز توأم
- بخش‌بندی و نگاشت
- الگوریتم‌های مکاشفه‌ای با استراتژی بهینه‌سازی
- Cosyma
- Global critically/Local phase



مباحث جلسه آینده

- فرایند سنتز توأم
- بخش‌بندی و نگاشت
- الگوریتم‌های بخش‌بندی در سیستم‌های توزیع شده