المحاضرة 1

بالطبع، إليك الترجمة الكاملة للنص إلى اللغة العربية مع الحفاظ على المعنى والمحتوى العلمي:

مقدمة في أساسيات الحاسوب

- يُبنى مقرر "بنية الحاسوب 2" على مفاهيم مقرّر "بنية الحاسوب 1"، حيث يتم مراجعة الأساسيات الضرورية لفهم بنية الحاسوب، لتكون قاعدة للمواضيع المتقدمة لاحقًا.
- تشمل المواضيع الرئيسية تصميم وحدة التحكم (بطريقتيها العتادية والبرمجية)، تطبيقات البرمجة المصغّرة (Microprogramming)، المعالجات المتزامنة، المعالجات فائقة التدرج (Superscalar)، الذاكرة الافتراضية، ومبادئ الحوسبة المتوازية مع أمثلة عليها.

بنية الحاسوب مقابل تنظيم الحاسوب

- **بنية الحاسوب (Computer Architecture):** تشير إلى الخصائص المرئية للنظام التي تؤثر على تنفيذ البرامج، كعرض البتات المستخدمة في تمثيل البيانات، آليات الإدخال والإخراج، أنماط الفهرسة، وطرق التعامل مع الذاكرة.
- **تنظيم الحاسوب (Computer Organization):** يتناول الوحدات التشغيلية والترابطات بينها التي تُحقّق البنية المعمارية، ويشمل تفاصيل العتاد التي تكون غالبًا غير مرئية للمبرمج، مثل إشارات التحكم، واجهات الأجهزة الطرفية، وتقنيات الذاكرة.

المكونات الأساسية للحاسوب (بنية فون نيومان)

- تتكوّن البنية الأساسية للحاسوب من ثلاث وحدات رئيسية: وحدة المعالجة المركزية (CPU)، وحدة الذاكرة الرئيسية، وأجهزة الإدخال/الإخراج.
- تتضمّن وحدة المعالجة المركزية وحدة التحكم (CU) ووحدة الحساب والمنطق (ALU)، المسؤولة عن التحكم في التعليمات ومعالجة البيانات على الترتيب.
- المهام الرئيسية للحاسوب هي: تخزين البيانات، نقل البيانات، معالجة البيانات، والتحكم – وكل مهمة ترتبط بمكون معين من مكونات الحاسوب.

العناصر العتادية الرئيسية

- **الترانزستور:** مفتاح كهربائي يمكن تشغيله وإيقافه، ويعدّ اللبنة الأساسية للدوائر الحاسوبية.
- **الدائرة المتكاملة (IC):** تجمع عدّة ترانزستورات ضمن شريحة واحدة، ما يسمح بتصميم عتاد مضغوط وفعّال.
- **الدمج واسع النطاق جدًا (VLSI):** دوائر تحتوي على مئات الآلاف إلى ملايين الترانزستورات، وتشكّل أساس المعالجات الحديثة المعقّدة.
- **السيليكون:** عنصر شبه موصل يُعالج كيميائيًا ليُستخدم في إنتاج الموصلات والعوازل والمفاتيح (الترانزستورات) داخل الشرائح.
- تُصنّف الدوائر المتكاملة حسب عدد الترانزستورات:
  - MSI (دمج متوسط): من 10 إلى 100 ترانزستور

- LSI (دمج واسع): من 100 إلى 1000 ترانزستور
- VLSI (دمج واسع جدًا): أكثر من 1000 ترانزستور

مستويات لغات البرمجة وعملية الترجمة

- توجد لغات البرمجة ضمن مستويات مختلفة:

1. **اللغات عالية المستوى:** أوامر مفهومة للبشر (مثل C وJava)، لا يفهمها الحاسوب مباشرة.
2. **لغة التجميع (Assembly):** لغة رمزية وسيطة تُترجم بواسطة المجمع (Assembler) إلى لغة الآلة؛ تعمل على المسجلات بدلاً من المتغيرات.
3. **لغة الآلة:** تتكوّن من شيفرات ثنائية (0 و1) تُنفّذ مباشرة من قبل عتاد الحاسوب.

- تتضمن عملية الترجمة استخدام المترجمات لتحويل الشيفرة عالية المستوى إلى شيفرة تجميعية أو وسيطة، ثم يتم تحويلها إلى لغة الآلة بواسطة المجمع.

وحدات تخزين الذاكرة

- تُقاس سعات الذاكرة وفق تدرج هرمي مبني على قوى العدد 1024:
  - 8 بت = 1 بايت
  - 1024 بايت = 1 كيلوبايت (KB)
  - 1024 كيلوبايت = 1 ميغابايت (MB)
  - 1024 ميغابايت = 1 غيغابايت (GB)
  - 1024 غيغابايت = 1 تيرابايت (TB)
- تُصنَع شرائح الحاسوب من سبائك السيليكون وأقراص السيليكون (wafers)، التي تُعالَج لتكوين الترانزستورات، الموصلات، والعوازل ضمن الدوائر المتكاملة.

بنية مجموعة التعليمات (ISA)

- مجموعة التعليمات هي اللغة الوحيدة التي يفهمها المعالج، وتمثّل الجسر بين البرامج والعتاد، وتختلف من معالج لآخر.
- من الأمثلة على البنى: RISC ،MIPS، وARM.
- تُصنّف تعليمات MIPS إلى:
  - **تعليمات R-Type:** للعمليات الحسابية والمنطقية، تتطلب ثلاث مسجلات (اثنان للمصدر وواحد للنتيجة)، وتستخدم وحدة الحساب والمنطق (ALU).
  - **تعليمات I-Type:** لتعليمات التعامل مع الذاكرة (مثل التحميل والتخزين)، وتُحسب العناوين الفعالة عبر جمع قيمة مباشرة بمحتوى مسجل.

- **تعليمات J-Type:** تعليمات القفز (Jump)، تغيّر عداد البرنامج (PC) باستخدام عنوان مكوّن من 26 بت يُعدّل ليصبح 32 بتًا، تشمل القفزات غير المشروطة والتفرعات الشرطية.

التسلسل الهرمي للذاكرة

- تُنظم الذاكرة بطريقة هرمية: كلما اقتربت من المعالج كانت أسرع لكن أصغر، وكلما ابتعدت كانت أبطأ لكنها أكبر.
- هذا التنظيم يوفّر موازنة بين السرعة والسعة، ويمنح المعالج "وهم" توفر ذاكرة كبيرة وسريعة.

العوامل المؤثرة في أداء البرنامج

- يتأثر أداء البرنامج بعدة عوامل: الخوارزمية، لغة البرمجة، كفاءة المترجم، بنية التعليمات، وبنية العتاد.
- من العوامل المؤثرة في أداء وحدة المعالجة المركزية:
  - عدد التعليمات المنفذة
  - عدد الدورات لكل تعليمة (CPI)
  - زمن دورة الساعة أو ترددها
- معادلة زمن تنفيذ المعالج:
  زمن CPU=عدد التعليمات×CPI×زمن دورة الساعةtext{زمن text = {CPU{عدد التعليمات} \times \text{CPI} \times \text{زمن دورة الساعة}
  أو بشكل آخر:
  زمن CPU=عدد التعليمات×CPIتردد الساعةtext{زمن text}frac = {CPU{عدد التعليمات} \text}}{\text{CPI}times \text{تردد الساعة}}}
- الـ CPI هو متوسط عدد دورات الساعة لكل تعليمة، ويعكس مدى تعقيد التعليمات وكفاءة العتاد.

الساعة ومقاييس الأداء

- **تردد الساعة (Clock Rate):** عدد دورات الساعة في الثانية، ويُقاس بالهرتز (Hz).
- **زمن دورة الساعة:** هو الوقت اللازم لدورة واحدة، والعلاقة بينهما هي:
  {f=1tf = \frac{1}{t}
  حيث ff هو التردد، وtt هو زمن الدورة.
- **الأداء (Performance):** هو مقلوب زمن التنفيذ:
  الأداء=1زمن التنفيذtext{الأداء} = \text}{1}frac{زمن التنفيذ}}
- يشمل زمن التنفيذ كل نشاطات النظام: تنفيذ المعالج، الوصول إلى الذاكرة، الإدخال/الإخراج، والتحميل الناتج من نظام التشغيل.

مثال: تحسين أداء المعالج

- إذا كان برنامج يُنفذ خلال 10 ثوانٍ على معالج بتردد 2 غيغاهرتز (الحاسوب A)، ونريد تنفيذه في 6 ثوانٍ على حاسوب B، والذي يحتاج إلى عدد دورات أكبر بـ1.2 مرة، فإن تردد الساعة اللازم لحاسوب B يُحسب كما يلي:

- عدد الدورات على A:
  $A = 10 \times 2 \times 10^9 = 20 \times 10^9$ الدورات$\text{الدورات}_{A}=10\times2\times109=20\times109$
- زمن التنفيذ على B:
  $B6 = \frac{1.2 \times 20 \times 10^9}{\text{ترددات}}$ $6=1.2\times20\times109$
- بحل المعادلة نجد:
  $B = \frac{1.2 \times 20 \times 10^9}{6} = 4$ ترددات_{B} $B=1.2\times20\times1096=4\times109=4$ ترددات
  $\times 10^9 = 4 \text{ غيغاهرتز}$
- بالتالي، يجب أن يكون تردد الساعة في الحاسوب B ضعف تردد الحاسوب A لتحقيق الهدف من الأداء.

هل ترغب أيضًا بملف Word أو PDF لهذا المحتوى؟

المحاضرة 2

... بالطبع، إليك الترجمة الكاملة للنص مع الحفاظ على المعنى والمحتوى العلمي:

تشغيل وحدة التحكم والعمليات الصغرية (Micro-Operations)

- **تتحمل وحدة التحكم (CU) المسؤولية الأساسية في توليد سلسلة من العمليات الصغرية خلال دورة تنفيذ التعليمة للتحكم في عمل وحدة المعالجة المركزية (CPU).**
- **العمليات الصغرية هي عمليات بسيطة جدًا تنفذها وحدة المعالجة المركزية، وتشكّل مراحل تنفيذ التعليمة مثل: الجلب (Fetch)، العنوان غير المباشر (Indirect Addressing)، التنفيذ (Execute)، والتعامل مع المقاطعات (Interrupt).**
- **تتكوّن كل دورة تعليمة من عدة عمليات صغرية تتحكم في سجلات المعالج ومسارات البيانات لتنفيذ وظيفة التعليمة.**
- **فهم العمليات الصغرية أساسي لتصميم وحدة التحكم وتوصيف تسلسل الأحداث داخل أي دورة تعليمة.**

دورة التعليمة ومراحلها

- **تتكوّن دورة التعليمة من أربع مراحل رئيسية:**

1. **الجلب (Fetch)**
2. **العنوان غير المباشر (إن وُجد)**
3. **التنفيذ (Execute)**

4. **التعامل مع المقاطعة (Interrupt)**

- **في مرحلة الجلب، تُجلب التعليمة من الذاكرة، ويُستخدم فيها عدة سجلات مثل:**

  1. **عداد البرنامج (PC)**
  2. **سجل عنوان الذاكرة (MAR)**
  3. **سجل مخزن الذاكرة (MBR)**
  4. **سجل التعليمة (IR)**

- **تسلسل الجلب يتضمن:**

  1. **وضع عنوان التعليمة التالية من PC في MAR.**
  2. **قراءة محتوى الذاكرة في MAR إلى MBR.**
  3. **زيادة قيمة PC بطول التعليمة.**
  4. **نقل التعليمة من MBR إلى IR.**

- **يتم ترتيب العمليات الصغرية في مرحلة الجلب بعناية لتجنب التعارضات، مثل محاولة القراءة والكتابة على نفس السجل في الوقت نفسه.**

دورة العنوان غير المباشر

- **إذا استخدمت التعليمة عنوانًا غير مباشر، تُضاف دورة إضافية قبل التنفيذ لجلب العنوان الفعّال من الذاكرة.**
- **تتضمّن هذه الدورة نقل حقل العنوان من IR إلى MAR، ثم قراءة محتوى الذاكرة إلى MBR، وأخيرًا تحديث عنوان IR بالقيمة الموجودة في MBR، مما يجعل التعليمة تبدو كما لو أنها استخدمت عنوانًا مباشرًا.**

دورة المقاطعة (Interrupt)

- **عند حدوث مقاطعة، تحفظ وحدة المعالجة القيمة الحالية لـ PC في الذاكرة، وتحمّل عنوان روتين خدمة المقاطعة (ISR) في PC.**
- **تتضمّن دورة المقاطعة عمليات صغرية لنقل PC إلى MBR، حفظ العنوان في الذاكرة، وتحديث PC بعنوان ISR لتمكين المعالج من التعامل مع المقاطعة قبل استئناف التنفيذ الطبيعي.**
- **تُستخدم أقنعة المقاطعة (Interrupt Masking) لمنع مقاطعة البرامج الحساسة، وذلك عن طريق تعطيل المقاطعات مؤقتًا لضمان تنفيذ التعليمات الهامة دون انقطاع.**

دورة التنفيذ وأمثلة

- تختلف دورة التنفيذ حسب رمز التعليمة (Opcode)، حيث أن لكل رمز تسلسلًا خاصًا من العمليات الصغرية.
- مثال 1: تعليمة ADD (جمع قيمة من الذاكرة مع سجل):
  - MAR ← عنوان من IR
  - MBR ← القيمة من Memory[MAR]
  - R1 ← R1 + MBR (باستخدام ALU)
- مثال 2: تعليمة ISZ (زيادة وتخطي إن كانت النتيجة صفر):
  - MAR ← عنوان من IR
  - MBR ← Memory[MAR]
  - MBR ← MBR + 1
  - Memory[MAR] ← MBR
  - إذا كانت MBR = 0 → PC ← PC + طول التعليمة
- مثال 3: تعليمة BSA (تخزين عنوان الرجوع والانتقال إلى روتين فرعي):
  - MAR ← عنوان من IR
  - MBR ← PC
  - Memory[MAR] ← MBR
  - PC ← عنوان من IR
  - PC ← PC + 1

وظائف وحدة التحكم وإشارات التحكم

- تؤدي وحدة التحكم وظيفتين أساسيتين:

1. التسلسل (Sequencing): توجيه المعالج عبر التسلسل الصحيح للعمليات الصغرية.
2. التنفيذ (Execution): توليد إشارات تحكم لتنفيذ كل عملية صغرية.

- **تُولّد إشارات التحكم بناءً على الحالة الحالية والتعليمة، وتُستخدم للتحكم في مسارات البيانات، والسجلات، ووحدة ALU.**
- **تتطلب وحدة التحكم مدخلات لمعرفة حالة النظام ومخرجات للتحكم في سلوك النظام، بما في ذلك تفعيل البوابات والنواقل لنقل البيانات.**

أنواع العمليات الصغرية

تصنّف العمليات الصغرية إلى:

1. **نقل البيانات بين السجلات.**
2. **نقل البيانات بين السجلات والأجهزة الخارجية (الذاكرة، الإدخال/الإخراج، ناقل النظام).**
3. **العمليات الحسابية أو المنطقية التي تنفذها ALU.**

التنظيم الداخلي ومسارات البيانات

- **غالبًا ما يُستخدم ناقل داخلي واحد تتحكم به بوابات لنقل البيانات بين السجلات وALU.**
- **تُحكم إشارات التحكم عملية النقل إلى ومن الناقل الداخلي والنواقل الخارجية.**
- **تُستخدم سجلات مؤقتة لحفظ البيانات الوسيطة أثناء تنفيذ عمليات ALU لضمان المعالجة الصحيحة.**

جدول موجز لأهم السجلات في دورة التعليمة

اسم السجل	الوظيفة
عداد البرنامج (PC)	يحتفظ بعنوان التعليمة التالية التي يجب جلبها
سجل عنوان الذاكرة (MAR)	يحتفظ بعنوان الذاكرة لعمليات القراءة/الكتابة
سجل مخزن الذاكرة (MBR)	يحتفظ بالبيانات المقروءة من الذاكرة أو التي ستُكتب إليها
سجل التعليمة (IR)	يحتفظ بالتعليمة الحالية التي يتم تنفيذها

🧩 مخطط التدفق

يوضح تسلسل العمليات الصغرية خلال دورة التعليمة، بما في ذلك الجلب، العنوان غير المباشر، التعامل مع المقاطعة، والتنفيذ.

❗ **هام: يجب ترتيب العمليات الصغرية بعناية لتجنب التعارضات مثل القراءة والكتابة على نفس السجل في الوقت نفسه، مما يضمن عمل وحدة المعالجة بثبات وموثوقية.**

💡 **معلومة أساسية: قدرة وحدة التحكم على توليد إشارات دقيقة للتحكم بالتسلسل والتنفيذ تُعدّ جوهر الأداء الصحيح لوحدة المعالجة المركزية (CPU).**

هل ترغب أن أدمج هذا المحتوى في عرض شرائح أو ملف PDF؟

المحاضرة 3

.. إليك **ملخصًا لمحتوى ملف "بنيان نظري 3"** الذي يتناول موضوع تصميم وحدة التحكم (Control Unit Design) ضمن مقرر بنية الحاسوب:

🧠 مقدمة عامة

- يوضح الملف الفرق بين **العتاديات الصلبة (Hardware)** مثل المعالج واللوحة الأم، و**البرمجيات (Software)** بنوعيها:
  1. تطبيقية مثل Office.
  2. نظامية مثل أنظمة التشغيل (Windows, Linux).
- يشرح المعالج على أنه يتكون من جزئين أساسيين:
  1. **Data Path**: مسار البيانات ويشمل الذاكرة والمبدلات.
  2. **Control Unit (CU)**: وحدة التحكم المسؤولة عن تحريك البيانات والتحكم بالتعليمات.

⚙️ مفاهيم أساسية

- **وحدة الحساب والمنطق (ALU)** مسؤولة عن تنفيذ العمليات الحسابية والمنطقية.
- المعالجات يتم تصميمها لتعيد استخدام أجزاء الـ Data Path المشترك بين التعليمات لتقليل التكرار.
- التتابع في عمل النظام يتم شرحه باستخدام **Finite State Machines (FSM)** (آلات الحالة المنتهية).

🔄 تصميم وحدة التحكم باستخدام FSM

1. ترميز الحالات (Encoding).
2. تصميم جدول الحقيقة باستخدام الحالة الحالية والدخل.
3. اشتقاق علاقات الحالة التالية والمخرجات.
4. تمثيل الحالات باستخدام القلابات (Flip-Flops) مثل JK.
5. استخدام جداول كارنوف لاشتقاق العلاقات المنطقية المبسطة.

💡 تصميم وحدة التحكم للمعالج MIPS

- المعالج MIPS هو من نوع **RISC** ويتميز ببساطة التعليمات مقارنة بـ Intel (CISC).
- هناك ثلاثة أنواع من التعليمات:
  1. **R-Type**: حسابية وتستخدم سجلات فقط.
  2. **I-Type**: تتضمن القيم المباشرة (Immediate).
  3. **J-Type**: نادرًا ما نوقشت.

🗒️ مكونات تعليمة MIPS

- الـ **Opcode** يحدد نوع التعليمة.
- الحقول مثل `rs`, `rt`, `rd` تشير إلى السجلات.
- قيمة الإزاحة (offset) تستخدم في تعليمات Branch و Load/Store.
- التصميم يتم على مرحلتين:
  1. **Main Control Unit** تولد إشارات تحكم أولية.
  2. **ALU Control Unit** تحدد نوع العملية الحسابية.

🧮 تحكم متعدد المستويات

- تقنيات Single-Cycle مقابل Multi-Cycle:
  - **Single-Cycle**: تنفذ كل تعليمة بدورة واحدة.
  - **Multi-Cycle**: تقسيم تنفيذ التعليمة إلى مراحل على عدة دورات.

- التحكم متعدد الدورات يستخدم FSM أكثر تعقيدًا.

🛠️ بدائل لتصميم وحدة التحكم

1. **المنطق العشوائي (Random Logic):** سريع لكن غير مرن.
2. **Firmware (البرامج الثابتة):**
   - تعتمد على تعليمات صغرية (Micro-instructions).
   - مرنة ويمكن تعديلها.
   - بطيئة مقارنة بالمنطق الصلب لأنها تعتمد على ذاكرة.

✅ أهم المفاهيم المغطاة

- التمثيل بالحالات FSM.
- تصميم وحدات التحكم باستخدام قلابات JK.
- تصميم جداول الحقيقة وجداول الانتقال.
- الفروقات بين أنواع التعليمات في MIPS.
- الفرق بين وحدات التحكم السلكية والمصغّرة.

إذا أردت تحويل هذا الملخص إلى **PDF** أو **Word** أو تلخيصًا بصريًا (مثلاً: خريطة ذهنية أو جدول مقارنة)، فأعلمني فقط.

المحاضرة 6 و 4 :

تمام، سأقدّم لك الملخص المفصّل بما يعادل تقريبًا 8 صفحات نصية، مع الحفاظ على التسلسل العلمي.

ملخص موسع – تصميم وحدة التحكم (Control Unit Design)

1. مقدمة

تعد **وحدة التحكم (Control Unit - CU)** من المكونات الأساسية لوحدة المعالجة المركزية (CPU)، إذ تتحمل مسؤولية إدارة وتنظيم سير العمليات داخل المعالج.
تعمل وحدة التحكم على **توليد إشارات التحكم** التي تحدد كيفية انتقال البيانات بين المكونات المختلفة وتنفيذ التعليمات.
وتتم هذه العملية عبر **العمليات الصغرية (Micro-operations)**، وهي خطوات صغيرة جدًا يتم تنفيذها على مستوى السجلات وناقل البيانات لضمان سير العمل بدقة وسرعة.

---

2. دورة التعليمة (Instruction Cycle)

تنقسم دورة التعليمة إلى أربع مراحل رئيسية:

1. **مرحلة الجلب (Fetch Cycle):**
   جلب التعليمة من الذاكرة إلى المعالج.

2. **مرحلة العنوان غير المباشر (Indirect Addressing Cycle)** – إذا كانت التعليمة تستخدم عنوانًا غير مباشر.

3. **مرحلة التنفيذ (Execute Cycle):**
   تنفيذ العملية المطلوبة بحسب Opcode التعليمة.

4. **مرحلة المقاطعة (Interrupt Cycle):**
   التعامل مع الأحداث الطارئة أو طلبات الأجهزة الطرفية.

تتكرر هذه الدورة بشكل مستمر أثناء تشغيل المعالج.

---

3. مرحلة الجلب (Fetch Cycle)

في هذه المرحلة:

1. نقل عنوان التعليمة من **عداد البرنامج (PC)** إلى **سجل عنوان الذاكرة (MAR)**.

2. قراءة محتوى الذاكرة عند هذا العنوان إلى **سجل مخزن الذاكرة (MBR)**.

3. زيادة قيمة **PC** بمقدار طول التعليمة.

4. نقل التعليمة من **MBR** إلى **سجل التعليمة (IR)**.

⚠️ يجب ترتيب العمليات الصغرية لتجنب التعارض، مثل القراءة والكتابة على نفس السجل في نفس اللحظة.

---

4. مرحلة العنوان غير المباشر (Indirect Addressing)

إذا كانت التعليمة تستخدم عنوانًا غير مباشر:

- يتم جلب العنوان الفعّال من الذاكرة قبل التنفيذ.
- تتم العملية عبر:

1. نقل حقل العنوان من IR إلى MAR.
2. قراءة محتوى الذاكرة إلى MBR.
3. تحديث IR بعنوان MBR.

بهذا تصبح التعليمة كما لو كانت تستخدم عنوانًا مباشرًا.

---

5. مرحلة التنفيذ (Execute Cycle)

تختلف هذه المرحلة حسب **Opcode** التعليمة.
أمثلة:

- **ADD:** جمع قيمة من موقع ذاكرة مع سجل.
  - MAR ← عنوان من IR
  - MBR ← Memory[MAR]
  - R1 ← R1 + MBR
- **ISZ:** زيادة موقع ذاكرة وتخطي التعليمة التالية إذا كانت النتيجة صفر.
  - MAR ← عنوان من IR
  - MBR ← Memory[MAR]
  - MBR ← MBR + 1
  - Memory[MAR] ← MBR
  - إذا كانت MBR = 0 → PC ← PC + طول التعليمة
- **BSA:** حفظ عنوان الرجوع والانتقال إلى روتين فرعي.

- MAR ← عنوان من IR
- MBR ← PC
- Memory[MAR] ← MBR
- PC ← عنوان من IR
- PC ← PC + 1

---

6. مرحلة المقاطعة (Interrupt Cycle)

عند حدوث مقاطعة:

- يتم حفظ قيمة **PC** الحالية في الذاكرة.
- تحميل عنوان **روتين خدمة المقاطعة (ISR)** في **PC**.
- تنفيذ إجراءات المعالجة، ثم العودة لإكمال البرنامج.

يمكن استخدام **أقنعة المقاطعة (Interrupt Masking)** لتعطيل المقاطعات مؤقتًا وحماية التعليمات الحساسة من الانقطاع.

---

7. وظائف وحدة التحكم

تنقسم وظائف وحدة التحكم إلى وظيفتين أساسيتين:

1. **التسلسل (Sequencing):** ضمان تنفيذ العمليات الصغرية بالترتيب الصحيح.
2. **التنفيذ (Execution):** توليد إشارات التحكم اللازمة لنقل البيانات وتشغيل ALU.

مدخلات وحدة التحكم:

- حالة النظام.
- التعليمة الحالية.
- إشارات التوقيت.

مخرجات وحدة التحكم:

- إشارات التحكم إلى المكونات الداخلية.
- إشارات تحكم إلى الناقلات الخارجية.

---

8. أنواع العمليات الصغرية (Micro-operations)

1. **نقل البيانات بين السجلات الداخلية.**
2. **نقل البيانات بين السجلات والذاكرة أو وحدات الإدخال/الإخراج.**
3. **تنفيذ عمليات حسابية أو منطقية في ALU.**

---

9. التنظيم الداخلي ومسارات البيانات

- غالبًا يوجد **ناقل داخلي واحد** يتصل بجميع السجلات ووحدة ALU.
- **بوابات التحكم** تستخدم لتوجيه البيانات عبر الناقل.
- توجد **سجلات مؤقتة** لتخزين القيم الوسيطة أثناء العمليات.

---

10. السجلات الأساسية في دورة التعليمة

السجل	الوظيفة
PC	يحتفظ بعنوان التعليمة التالية
MAR	يحتفظ بعنوان الذاكرة الجاري الوصول إليه
MBR	يحتفظ بالبيانات المقروءة من أو المكتوبة إلى الذاكرة
IR	يحتفظ بالتعليمة الحالية

---

11. تصميم وحدة التحكم

أنواع التصميم:

1. **التحكم السلكي (Hardwired Control):**
   - سريع.
   - أقل مرونة في التعديل.
2. **التحكم المصغر (Microprogrammed Control):**
   - مرن وسهل التطوير.
   - أبطأ من التحكم السلكي.

آلية التحكم المصغر:

- تخزين تعليمات صغرية في ذاكرة خاصة (Control Memory).
- قراءتها لتوليد إشارات التحكم.

استخدام FSM (Finite State Machine):

- تمثيل حالات التنفيذ وتسلسل الانتقال بينها.

---

12. الخلاصة

- وحدة التحكم هي المحرك الأساسي لعمل المعالج.
- تتطلب معرفة دقيقة بدورة التعليمة والعمليات الصغرية.
- اختيار تصميم وحدة التحكم يعتمد على التوازن بين الأداء والمرونة.

إذا أردت يمكنني الآن أن أحوّل هذا الملخص إلى **مخطط بصري (Diagram)** يوضح المراحل والعمليات الصغرية ليسهُل عليك الحفظ والمراجعة.
هل تريد أن أعدّه لك؟

ملخص اصغر ل 6 و 4

.....

سأقوم بإعداد ملخص منظم في 4 صفحات لمحتوى ملفك، مع الحفاظ على النقاط العلمية الأساسية وترتيب الأفكار ليسهُل المراجعة.

ملخص محتوى الملف – تصميم وحدة التحكم (Control Unit Design)

1. مقدمة

- وحدة التحكم (CU) هي المكوّن المسؤول عن توجيه عمل المعالج من خلال توليد إشارات تحكم تنسّق نقل البيانات وتنفيذ التعليمات.
- يتم التنفيذ عبر عمليات صغرية (Micro-operations)، وهي خطوات صغيرة جدًا تتحكم في السجلات ومسارات البيانات.
- دورة التعليمة تشمل عادة:

1. الجلب (Fetch)
2. العنوان غير المباشر (Indirect Addressing) – إذا لزم الأمر
3. التنفيذ (Execute)
4. التعامل مع المقاطعة (Interrupt)

---

2. العمليات الصغرية ودورة التعليمة

- العمليات الصغرية: نقل بيانات بين السجلات، عمليات ALU، أو عمليات إدخال/إخراج.
- مرحلة الجلب:
  - MAR ← PC
  - [MBR ← Memory[MAR
  - PC ← PC + طول التعليمة
  - IR ← MBR
- يجب ترتيب العمليات لتجنب التعارضات مثل القراءة والكتابة على نفس السجل.
- مرحلة العنوان غير المباشر:
  - إذا استخدمت التعليمة عنوانًا غير مباشر، يُجلب العنوان الفعلي من الذاكرة قبل التنفيذ.
- مرحلة التنفيذ:
  - تختلف باختلاف Opcode التعليمة.
  - أمثلة:
    - ADD: جمع محتوى الذاكرة مع سجل.
    - ISZ: زيادة قيمة موقع ذاكرة وتخطي التعليمة التالية إذا كانت النتيجة صفر.
    - BSA: حفظ عنوان الرجوع والانتقال إلى روتين فرعي.

- مرحلة المقاطعة:
  - حفظ PC في الذاكرة، تحميل عنوان روتين خدمة المقاطعة في PC، ومعالجة المقاطعة قبل استئناف البرنامج.
  - أقنعة المقاطعة تمنع مقاطعة البرامج الحرجة.

---

3. وظائف وحدة التحكم وإشارات التحكم

- التسلسل (Sequencing): ضمان ترتيب صحيح للعمليات الصغرية.
- التنفيذ (Execution): توليد إشارات تحكم لتحريك البيانات وتشغيل ALU.
- إشارات التحكم تتحكم في:
  - تفعيل البوابات والنواقل.
  - قراءة/كتابة السجلات.
  - اختيار عملية ALU.
- مدخلات وحدة التحكم: حالة النظام، التعليمة، إشارات التوقيت.
- مخرجات وحدة التحكم: إشارات التحكم نحو المعالج والنواقل.

---

4. أنواع العمليات الصغرية

1. نقل بيانات بين السجلات الداخلية.
2. نقل بيانات بين السجلات والذاكرة أو وحدات الإدخال/الإخراج.
3. تنفيذ عمليات حسابية أو منطقية في ALU.

---

5. التنظيم الداخلي ومسارات البيانات

- وجود ناقل داخلي واحد غالبًا، مع بوابات تحكم تتحكم بحركة البيانات.
- سجلات مؤقتة لتخزين القيم الوسيطة.
- إشارات التحكم تدير النقل الداخلي والخارجي.

---

6. السجلات الأساسية في دورة التعليمة

السجل	الوظيفة
PC	عنوان التعليمة التالية
MAR	عنوان الذاكرة لعمليات القراءة/الكتابة
MBR	البيانات المقروءة من أو المكتوبة إلى الذاكرة
IR	التعليمة الجاري تنفيذها

---

7. تصميم وحدة التحكم

- طرق التصميم:
  - التحكم السلكي (Hardwired): سريع لكن أقل مرونة.
  - التحكم المصغر (Microprogrammed): مرن وأسهل للتعديل لكنه أبطأ.
- آلية التحكم المصغر: تخزين تعليمات صغرية (Micro-instructions) في ذاكرة خاصة، وقراءتها لتوليد إشارات التحكم.
- FSM (آلة الحالة المنتهية):
  - تمثل حالات التنفيذ وتسلسل الانتقال بينها.
  - تُستخدم لتصميم وحدة التحكم سواء كانت سلكية أو مصغّرة.

---

8. خلاصة

- وحدة التحكم هي "المخرج" الذي يدير جميع أجزاء المعالج.
- تصميمها يتطلب فهم دقيق لدورة التعليمة، العمليات الصغرية، ومسارات البيانات.
- الأداء يعتمد على دقة إشارات التحكم وترتيب التنفيذ.
- اختيار نوع وحدة التحكم (سلكية أو مصغرة) يعتمد على التوازن بين السرعة والمرونة.

المحاضرة 5 :

سأقدّم لك الآن **ملخصًا موسعًا** لمحتوى الملف بحيث يعادل تقريبًا 8 صفحات نصية منظمة، مع الحفاظ على المحتوى العلمي والتسلسل المنطقي.

ملخص موسع – تصميم وحدة التحكم (Control Unit Design)

1. مقدمة

تعتبر **وحدة التحكم (CU)** في المعالج من أهم المكونات المسؤولة عن تنسيق العمليات داخل وحدة المعالجة المركزية (CPU).
وظيفتها الأساسية هي إصدار **إشارات التحكم** التي تحدد حركة البيانات وتنفيذ التعليمات وفقًا لمراحل دورة التعليمة.
تنفذ وحدة التحكم مهامها عبر **عمليات صغرية (Micro-operations)**، وهي عمليات أساسية يتم تنفيذها على السجلات وناقل البيانات.

---

2. إشارات التحكم الأساسية

يتطلب تشغيل وحدة التحكم استخدام إشارات تحكم رئيسية تشمل:

- **IorD**: اختيار مصدر العنوان (PC أو سجل عنوان الذاكرة).
- **MemRead / MemWrite**: التحكم في عمليات قراءة/كتابة الذاكرة.
- **IRWrite**: تحميل سجل التعليمات بالقيمة الجديدة.
- **ALUSrcA / ALUSrcB**: تحديد مصادر دخل ALU.
- **ALUOp**: تحديد العملية التي ستنفذها ALU.
- **PCSrc / PCWriteCond**: التحكم في تحديث قيمة PC.
- **RegDst / Mem2Reg / RegWrite**: التحكم في الكتابة على السجلات.

تنقسم الإشارات إلى نوعين:

1. **Single-bit control signals** (بت واحد): تشغيل/إيقاف وظيفة.

2. **Multi-bit signals** (متعددة البتات): اختيار مصدر أو عملية معينة.

---

3. مراحل دورة التعليمة (Instruction Cycle)

دورة التعليمة تتكون من **5 مراحل رئيسية**:

3.1 جلب التعليمة (Instruction Fetch – IF)

- قراءة التعليمة من الذاكرة إلى **IR**.
- تحديث **PC** بإضافة 4.

3.2 فك التشفير (Instruction Decode – ID)

- تحديد نوع التعليمة (R، I، J).
- قراءة البيانات اللازمة من السجلات.

3.3 التنفيذ (Execute – EX)

- **R-Type**: تنفيذ العملية بين السجلات.
- **I-Type**: حساب العناوين أو تنفيذ أوامر المقارنة.
- **J-Type**: تنفيذ القفزات.

3.4 الذاكرة (Memory)

- تنفيذ عمليات قراءة/كتابة الذاكرة إن وجدت.

3.5 الكتابة النهائية (Write Back)

- كتابة النتيجة إلى السجل الهدف أو تحديث PC.

---

4. تصميم التحكم متعدد الدورات

يعتمد هذا الأسلوب على تقسيم التعليمة إلى مراحل صغيرة بحيث تستخدم المكونات عدة مرات، مما يقلل من التعقيد المادي (Hardware Complexity) مقارنة بالتحكم أحادي الدورة.

يوجد أسلوبان أساسيان لتصميم وحدة التحكم:

1. **Hardwired Control**:
   - يعتمد على دوائر منطقية ثابتة.
   - سريع ولكنه أقل مرونة للتعديل.
2. **Micro-programmed Control**:
   - يعتمد على تخزين تعليمات صغرية في ذاكرة تحكم (Control Memory).
   - مرن وسهل التعديل لكنه أبطأ.

---

5. بدائل تصميم وحدة التحكم

- **Random Logic**: تصميم مخصص بالدوائر المنطقية.
- **PLA / ROM**: تخزين خرائط الإشارات في ذاكرة قابلة للبرمجة.
- **Micro-programming**: تنفيذ عمليات صغرية عبر تعليمات مخزنة في الذاكرة.

---

6. مثال ISA مبسط (غير MIPS)

يتناول المثال تعليمات مثل:

- **ADD x1**, **AND x1**, **JMP x1**, **INC**
  باستخدام سجلات مثل:
- **AR**: سجل العنوان.
- **DR**: سجل البيانات.
- **IR**: سجل التعليمات.

- **AC**: المراكم.

خطوات التصميم:

1. تقسيم التعليمات إلى مراحل فرعية (Add1، F3، F2، F1...).
2. رسم مخطط الحالة (State Diagram).
3. تصميم مسارات البيانات وتحديد الإشارات.
4. توليد الإشارات عبر ROM أو PLA.

---

7. البرمجة الصغرية (Microprogramming)

- **الفكرة**: تمثيل وحدة التحكم كمعالج صغير ينفذ تعليمات صغرية مخزنة في ذاكرة.
- **أنواع الترميز**:
  1. **Direct Encoding**: لكل إشارة تحكم بت منفصل.
  2. **Horizontal Encoding**: لكل عملية صغرية بت منفصل.
  3. **Vertical Encoding**: ترميز العمليات الصغرية برموز ثنائية لتقليل حجم الذاكرة.

---

8. حساب حجم الذاكرة المطلوبة

مثال لمعالج به:

- 120 إشارة تحكم.
- 63 عملية صغرية.
- 29 تعليمة صغرية.

حجم الذاكرة المطلوب:

- **مباشر**: 126 بت × 29 كلمة.
- **أفقي**: 69 بت × 29 كلمة.
- **عمودي**: 17 بت × 29 كلمة.

---

9. الخلاصة

- وحدة التحكم هي مركز التنسيق للمعالج.
- اختيار أسلوب التصميم يعتمد على التوازن بين السرعة والمرونة.
- البرمجة الصغرية توفر مرونة، والتحكم السلكي يوفر أداء أعلى.
- فهم دورة التعليمة وإشارات التحكم هو الأساس في تصميم أي معالج.

المحاضرة 7

إليك الترجمة الدقيقة للنص مع الحفاظ على التفاصيل العلمية والمصطلحات التقنية:

التوريد (Pipelining): تحسين أداء المعالج

- **التوريد (Pipelining)** هو تقنية لتحسين أداء المعالج عبر تنفيذ عدة تعليمات في وقت واحد بطريقة متسلسلة متداخلة.
- تُعد هذه الطريقة شكلًا من **التوازي على مستوى التعليمة (Instruction Level Parallelism – ILP)**، حيث يتم إدخال أكثر من تعليمة إلى المعالج في نبضات ساعة متتالية.
- يتم تقسيم مكوّنات العتاد إلى مراحل لتسهيل هذه النبضات، مع استخدام إشارات تحكم لإدارة تخزين القيم في السجلات لضمان توفرها للمراحل اللاحقة.

مراحل تنفيذ التعليمة

- تمر التعليمة عادةً بالمراحل التالية:

1. **جلب التعليمة (Instruction Fetch – IF)**: جلب التعليمة من **ذاكرة التعليمات (Instruction Memory – IM)**. في المعالجات أحادية الدورة، هناك ذاكرة منفصلة للتعليمات وأخرى للبيانات.
2. **فك التعليمة (Instruction Decode – ID)**: جلب القيم من السجلين `rs$` و`rt$` من **ملف السجلات (Register File)**.
3. **التنفيذ (Execute – EX)**: تنفيذ العمليات بواسطة **وحدة الحساب والمنطق (ALU)**.
4. **الوصول للذاكرة (Memory Access – MEM)**: ليست كل التعليمات تحتاجها، وتتم بعد حساب العنوان بواسطة ALU، باستخدام **ذاكرة بيانات** منفصلة عن ذاكرة التعليمات.
5. **الكتابة للخلف (Write Back – WB)**: كتابة النتائج إلى **ملف السجلات**.

مفهوم التوريد وتحسين الأداء

- يعتمد أداء المعالج على ثلاثة عوامل رئيسية: **عدد التعليمات، زمن دورة الساعة،** و**عدد الدورات لكل تعليمة (CPI)**.
- يزيد التوريد الأداء بشكل كبير عبر تنفيذ تعليمات متعددة في مراحل مختلفة بالتوازي، ما يرفع **معدل الإنتاجية (Throughput)**.
- يمكن أن يصل التسريع الناتج عن التوريد نظريًا إلى عدد المراحل في خط الأنابيب.
- يمكن أن تقل الكفاءة بسبب تفاوت أطوال المراحل أو التوقفات (Stalls) اللازمة لملء أو تفريغ خط الأنابيب.
- معادلة حساب التسريع:

Speedup=زمن المعالج القديمزمن المعالج الجديدعدد المراحلfrac\ = Speedup{زمن\ المعالج\ القديم}{زمن\ المعالج\ الجديد} \approx عدد\ المراحل

التوريد في معمارية MIPS

- تدعم معمارية MIPS التوريد بسهولة بسبب:
  - جميع التعليمات بنفس الطول، مما يبسط الجلب وفك التعليمات.
  - وجود عدد قليل من صيغ التعليمات، ما يسهل فكها في مرحلة واحدة.
  - أوامر الذاكرة تظهر فقط في تعليمات **التحميل والتخزين (Load/Store)**، ما يسمح بتأجيل الوصول للذاكرة لمرحلة لاحقة محددة.
  - البيانات في الذاكرة **مصطفة (Aligned)**، لذا تحتاج تعليمات نقل البيانات وصولًا واحدًا فقط للذاكرة.
- مثال: تعليمة `lw` (تحميل كلمة) تستغرق 8 نانوثانية في دورة واحدة، لكن باستخدام التوريد يمكن تقليص زمن دورة الساعة إلى 2 نانوثانية، محدودة بأطول مرحلة.

مخاطر خط الأنابيب (Pipeline Hazards)

- **الخطر (Hazard)** هو حالة تمنع تنفيذ التعليمة التالية في دورتها المحددة بسبب اعتماديات أو تعارض في الموارد، مما يعيق التدفق السلس للتعليمات.
- الأنواع الثلاثة:

1. **خطر بنيوي (Structural Hazard)**: يحدث عند محاولة مرحلتين مختلفتين استخدام نفس المورد في الوقت نفسه.

2. **خطر بيانات (Data Hazard)**: يحدث عندما تحتاج تعليمة إلى بيانات لم تُنتج بعد من تعليمة سابقة.

3. **خطر تحكم (Control Hazard)**: يحدث مع تعليمات القفز، عندما يعتمد جلب التعليمة التالية على نتيجة تعليمة لم تُنفذ بالكامل بعد.

حلول المخاطر البنيوية

- المشكلة الشائعة: استخدام وحدة ذاكرة واحدة لكل من جلب التعليمات والوصول للبيانات.
- الحلول:
  - **إضافة عتاد إضافي**، مثل وجود ملفي سجلات بدلًا من واحد.
  - **فصل وحدات الذاكرة** (تعليمات وبيانات منفصلتان) أو استخدام **ذاكرة ثنائية المنفذ (Dual-Port Memory)**.
  - إذا كان العتاد الإضافي غير ممكن، يتم إدخال **توقفات أو فقاعات (Bubbles)** عبر تعليمات **NOP** حتى يتاح المورد.
  - مثال: معالج به خطر بنيوي (40٪ تعليمات بيانات، معدل ساعة أعلى بـ 1.05 مرة) يمكن أن يكون أبطأ بنسبة 30٪ مقارنة بمعالج بدون هذا الخطر.
  - ملف السجلات في MIPS يدعم القراءة والكتابة في نفس دورة الساعة عبر تقسيم الدورة إلى نصفين.

حلول مخاطر البيانات

- تحدث عندما تحتاج تعليمة إلى بيانات لم تُكتب بعد في ملف السجلات.
- الحلول:
  - **التوقفات (Stalls)**: إدخال NOPs حتى تصبح البيانات متاحة.
  - **التجاوز (Forwarding / Bypassing)**: إرسال النتيجة مباشرة من مخرجات ALU أو الذاكرة إلى مدخلات ALU للتعليمة التالية دون انتظار الكتابة للسجلات.

- **مخاطر تحميل-استخدام (Load-Use Hazard)**: حالة خاصة حيث تحتاج التعليمة التالية مباشرةً بعد `load` إلى البيانات المحملة، مما قد يستلزم توقفًا حتى مع وجود التجاوز.

- **إعادة ترتيب التعليمات** بواسطة المترجم لملء الفجوات وتقليل الحاجة إلى التوقفات.

حلول مخاطر التحكم

- تحدث بسبب تعليمات القفز، إذ يحتاج المعالج لمعرفة نتيجة القفز وعنوان الهدف قبل جلب التعليمة التالية.

- الحلول:

- **التوقفات** حتى تُعرف نتيجة القفز.

- **القفز المؤجل (Delayed Branching)**: إعادة ترتيب التعليمات لوضع تعليمة مستقلة مباشرة بعد تعليمة القفز، بحيث تُنفذ دائمًا سواء تم القفز أم لا.

- **تنبؤ القفز (Branch Prediction)**: المعالج يتنبأ بنتيجة القفز ويواصل التنفيذ، ويعيد تشغيل خط الأنابيب إذا كان التنبؤ خاطئًا.

إذا رغبت، يمكنني الآن **دمج هذه الترجمة مع الجزء السابق الذي ترجمته لك** بحيث تحصل على ملف كامل ومترابط يغطي التوريد ومخاطره وحلولها. هل تريد أن أفعل ذلك؟

المحاضرة 9

بالطبع، إليك الترجمة الدقيقة مع الحفاظ على التفاصيل والمصطلحات العلمية:

مخاطر خط الأنابيب (Pipelining Hazards) والحلول

- تواجه المعالجات ذات خط الأنابيب ثلاثة أنواع من المخاطر: **مخاطر بنيوية (Structural Hazards)**، و**مخاطر بيانات (Data Hazards)**، و**مخاطر تحكم (Control Hazards)**.
- الحلول الشائعة تشمل إضافة عتاد إضافي (مثل زيادة الموارد أو استخدام تقنية **Forwarding**)، وإدخال **توقفات (Stalls)** أو تعليمات **NOP** (لا عملية)، أو إعادة ترتيب/جدولة شيفرة البرنامج.
- في معالجات **MIPS** القياسية، تظهر مشاكل خاصة من **مخاطر تحميل-استخدام البيانات (Load-use Data Hazards)** ومن عمليات **القفز (Branch)**.
- أي تحسين لعملية القفز يتطلب عادةً على الأقل توقفًا واحدًا (Stall).

أنواع الاعتماديات (Dependencies)

تعبر الاعتماديات عن العلاقات بين التعليمات التي يمكن أن تؤدي إلى مخاطر، وهناك ثلاثة أنواع رئيسية:

1. **اعتمادية البيانات الحقيقية (True Data Dependences)**
   - تحدث عندما تعتمد تعليمة على ناتج تعليمة سابقة، بحيث لا يمكن تنفيذ التعليمة الثانية قبل إتمام الأولى.
   - يحدث هذا إذا كان خرج التعليمة `i` هو دخل للتعليمة `j`، أو إذا كانت `j` تعتمد على `k` التي بدورها تعتمد على `i` (سلسلة اعتماديات).
   - يمكن أن تنتقل البيانات عبر السجلات (سهولة الكشف) أو عبر مواقع الذاكرة (صعوبة الكشف بسبب مشكلة **التحايل – Aliasing**).
2. **اعتمادية الأسماء (Name Dependences)**
   - تنشأ عندما تستخدم تعليمات نفس السجل أو الموقع في الذاكرة (الاسم نفسه) دون وجود تدفق بيانات مباشر بينهما.

- **اعتمادية عكسية (Anti-dependence)**: التعليمة `j` تكتب في موقع تقرأ منه التعليمة `i`.
- **اعتمادية إخراجية (Output dependence)**: التعليمتان `i` و `j` تكتبان في نفس الموقع، ويجب الحفاظ على ترتيب الكتابة.
- هذه ليست اعتماديات بيانات حقيقية، ويمكن حلها غالبًا عبر **إعادة تسمية السجلات (Register Renaming)** سواءً من قبل المترجم (ثابتًا) أو من قبل العتاد (ديناميكيًا).

3. **اعتمادية التحكم (Control Dependences)**

- تحدث عندما يعتمد تنفيذ تعليمة ما على نتيجة تعليمة تحكم سابقة (مثل: `jump`، `loop`، `else`، `if`، استدعاء دالة).
- تفرض اعتماديات التحكم قيودًا على إعادة ترتيب التعليمات:
  - لا يمكن نقل تعليمة تعتمد على فرع قبل هذا الفرع.
  - لا يمكن نقل تعليمة غير مرتبطة بفرع بعده إذا كان ذلك سيجعلها خاضعة له.

تصنيف مخاطر البيانات (Data Hazard Classification)

- **المخاطر (Hazards)** هي مشاكل تمنع التعليمة التالية في خط الأنابيب من التنفيذ في دورتها المخصصة.
- تحدث مخاطر البيانات عندما توجد اعتماديات بيانات أو أسماء بين تعليمات قريبة زمنيًا.

الأنواع:

1. **RAW (Read After Write)**: التعليمة `j` تحاول قراءة سجل قبل أن تكتب التعليمة `i` قيمته. (اعتمادية بيانات حقيقية).
2. **WAW (Write After Write)**: التعليمة `j` تحاول الكتابة إلى سجل قبل أن تكمل التعليمة `i` الكتابة إليه، مما يسبب خطأ في ترتيب الكتابة (اعتمادية إخراجية).
3. **WAR (Write After Read)**: التعليمة `j` تحاول الكتابة إلى سجل قبل أن تقرأه التعليمة `i`، ما قد يؤدي إلى قراءة قيمة خاطئة (اعتمادية عكسية – نادرة في خطوط الأنابيب التقليدية).
4. **RAR (Read After Read)**: ليس خطرًا، لأنه لا يغير القيم.

اعتماديات التحكم وإعادة الترتيب

- الحفاظ على بنية تعليمات التحكم ضروري لصحة البرنامج.
- يعتمد ذلك على:

1. **سلوك الاستثناءات (Exception Behavior)**: يجب ألا تؤدي إعادة الترتيب إلى استثناءات جديدة.
2. **تدفق البيانات (Data Flow)**: تعليمات القفز تجعل تدفق البيانات ديناميكيًا، لذا يحدد ترتيب البرنامج أي عملية توفر القيمة اللازمة.

- مثال: إعادة ترتيب `(lw R1, 0(R2` قبل `BEQZ R2, L1` قد تسبب **خطأ وصول غير قانوني للذاكرة** إذا كانت قيمة `R2` تساوي صفرًا.

التخمين (Speculation)

- التخمين يعني التنبؤ بنتيجة تعليمة (من قبل المترجم أو المعالج) لإزالة الاعتمادية والسماح بتنفيذ تعليمات لاحقة مبكرًا.
- قد يشمل:
  - التنبؤ بنتيجة القفز (Branch Prediction) لتنفيذ التعليمات التالية.
  - التنبؤ بأن تعليمة **store** قبل **load** لا تشير لنفس العنوان، ما يسمح بتنفيذ **load** أولًا.

أنواع التخمين:

1. **تخمين برمجي (Software Speculation)**: المترجم يعيد ترتيب التعليمات ويضيف تعليمات تحقق واستعادة عند الخطأ.
2. **تخمين عتادي (Hardware Speculation)**: المعالج نفسه يتنبأ ويخزن النتائج مؤقتًا، ويعتمدها إذا كان التنبؤ صحيحًا أو يلغيها إذا كان خاطئًا، باستخدام تقنيات مثل:
   - **التنبؤ الديناميكي بالفروع (Dynamic Branch Prediction)**.
   - **جدولة ديناميكية (Dynamic Scheduling)**.

- **ذاكرة إعادة الترتيب (Reorder Buffer – ROB)** ومرحلة الالتزام (Commit Stage).

التوازي على مستوى التعليمة (Instruction-Level Parallelism – ILP)

- يعني استغلال قدرة المعالج على تنفيذ تعليمات متعددة بالتوازي.
- طرق زيادة ILP:
  1. **زيادة عمق خط الأنابيب** (Pipeline Depth).
  2. **الإصدار المتعدد (Multiple Issuing)**: مضاعفة الوحدات الداخلية لإصدار أكثر من تعليمة في الدورة.

سياسات الإصدار (Issue Policies):

- **إصدار مرتب مع إنهاء مرتب (In-Order Issue, In-Order Completion)**.
- **إصدار مرتب مع إنهاء غير مرتب (In-Order Issue, Out-of-Order Completion)**.
- **إصدار غير مرتب مع إنهاء غير مرتب (Out-of-Order Issue, Out-of-Order Completion)**.

أنماط الإصدار:

- **إصدار ثابت (Static Multiple Issue)**: المترجم يحدد الحزم (Issue Packets) مسبقًا.
- **إصدار ديناميكي (Dynamic Multiple Issue)**: المعالج يحدد الحزم أثناء التشغيل (كما في المعالجات فائقة التدرج – Superscalar).

مثال: إصدار ثابت ثنائي في MIPS

- المعالج يصدر في كل دورة:
  - تعليمة ALU أو قفز واحدة.
  - وتعليمة تحميل/تخزين واحدة.

- يتطلب قراءة/كتابة سجلات متعددة في نفس الدورة، ومُجمِّع إضافي لحساب العناوين.
- معدل التأخير بين التحميل والاستخدام (Load-Use Latency) دورة واحدة.

إلغاء التكرار في الحلقات (Loop Unrolling)

- تقنية لتحسين **عدد الدورات لكل تعليمة (CPI)** عبر تقليل التوقفات وزيادة ILP.
- تتضمن:
  - إعادة ترتيب التعليمات لتقليل الاعتماديات.
  - تعديل الإزاحات في التعليمات.
  - استخدام **إعادة تسمية السجلات**.
- يمكن أن تقلل الـ CPI من 0.8 إلى 0.5 في حالات مثالية.

إذا أردت يمكنني أيضًا **إضافة الرسوم التوضيحية والمخططات** الخاصة بالاعتماديات والمخاطر لزيادة وضوح الشرح.
هل تريد أن أضيفها الآن؟