جامعة المنارة الخاصة
كلـــــية الهندســـــة
هندسة ميـــكاترونيك

المعالجات الصغرية ولغة التجميع

المحاضرة الخامسة

مدرس المقرر
د. بسام حسن

2025_2026

جَامعة المَنارة
MANARA UNIVERSITY

مفردات من المحاضرة الخامسة :

- Arithmetic and Logic Unit (ALU) Design

Arithmetic and Logic Unit (ALU) Design

البوابات المنطقية

- تحقق الوظائف المنطقية الأساسية
  - بوابة AND
  - بوابة OR
  - بوابة NOT
  - بوابة NAND
  - بوابة NOR
- يمكن تحقيق أي تابع منطقي باستخدام البوابة NOR أو باستخدام البوابة NAND

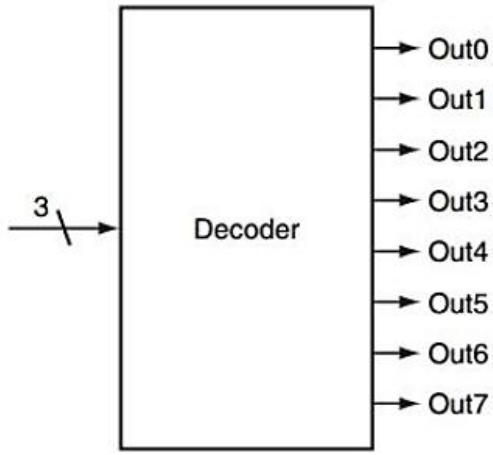
أصناف الكتل المنطقية Logic Blocks

- تصنف الكتل المنطقية Logic Blocks من حيث احتوائها على ذاكرة إلى نوعين
  - الكتل التركيبية Combinational
    - لا تحوي على ذاكرة
    - يتحدد الخرج فقط بقيم الدخل
    - ليس لها حالة (Stateless)
  - الكتل التتابعية Sequential Logic
    - تحوي ذاكرة
    - لها حالة (هي القيم المخزنة في الذاكرة)
    - يتحدد الخرج بقيم الدخل وبالحالة

المنطق التركيبي Combinational Logic

- مفككك الترميز Decoder
  - كتلة منطقية لها n بت دخل و 2^n بت خرج
  - يفعل واحد فقط من المخارج من أجل كل تركيبة من قيم الدخل
  - مثال: مفكك ترميز ثلاثي المداخل

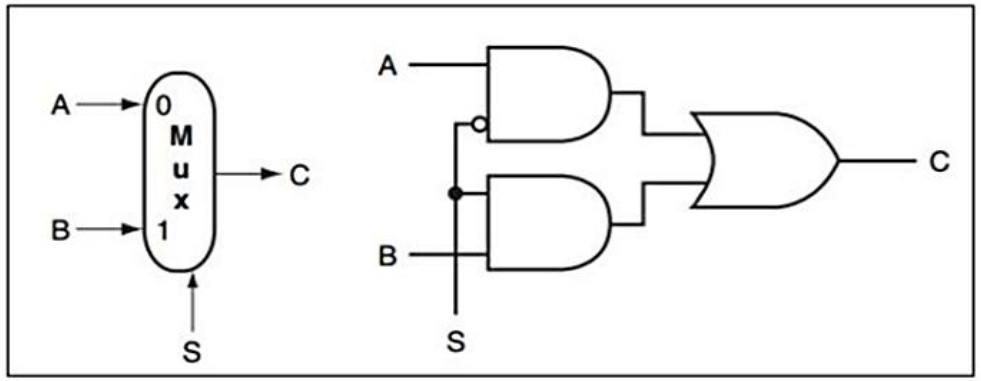
- مفككك الترميز Decoder
- الناخب Multiplexor/Selector



Inputs			Outputs							
12	11	10	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Multiplexor /Selector الناخب

- Multiplexor /Selector الناخب
  - كتلة منطقية لها
    - n مدخل معطيات
    - Log_2n مدخل اختيار (تحكم)
    - مخرج وحيد
  - يختار الناخب مدخل المعطيات الذي ستظهر قيمته على الخرج
  - وذلك: وفقا لقيمة مداخل الاختيار (تسمى قيمة التحكم Control Value)
- مثال: ناخب بسيط ثنائي المداخل
  - مدخل تحكم وحيد

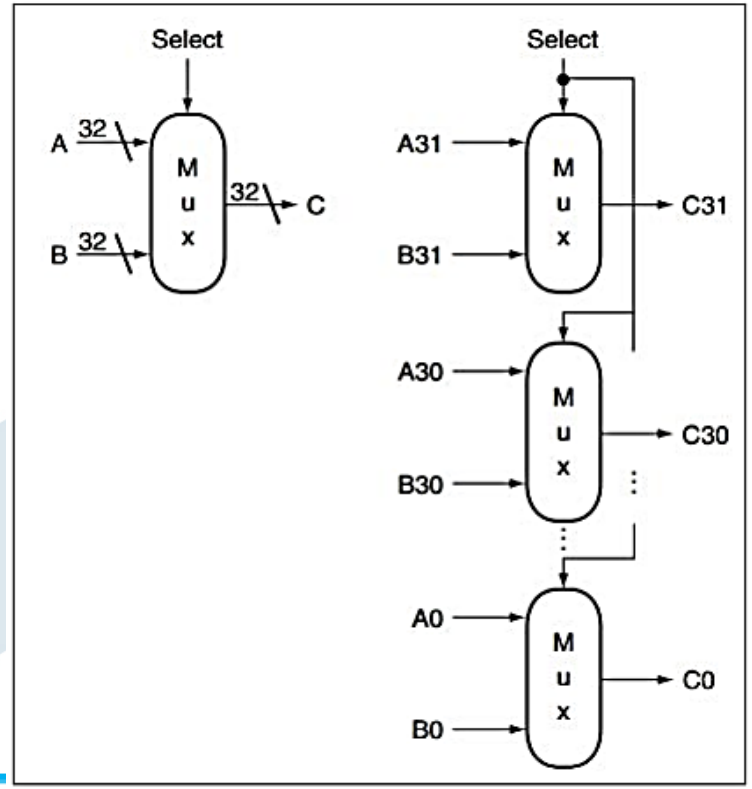


- في الحالة العامة يتألف الناخب من ثلاثة أقسام
  - مفكك ترميز: يولد n إشارة, تمثل كل واحدة منها اختيار واحد من المداخل
  - مصفوفة من n بوابة AND: تربط كل بوابة أحد المداخل مع أحد مخارج مفكك الترميز
  - بوابة OR واحدة: لها n مدخل

استخدام الناخب لاختيار القيم على ممر

Arithmetic and Logic Unit (ALU) Design

- الممر Bus
  - تعريف: الممر هو مجموعة من خطوط (المعطيات) تعامل باعتبارها إشارة منطقية واحدة
    - مثال: الخطوط التي تنقل قيمة من أحد المسجلات إلى الذاكرة (عرضها 32 bit)
    - هذا التعريف هو المناسب في سياق هذا العرض!
  - تعريف آخر: الممر هو مجموعة خطوط تأتي قيمها من مصادر مختلفة ولكل منها استخدام خاص
    - مثال: ممر الدخل/خرج I/O Bus



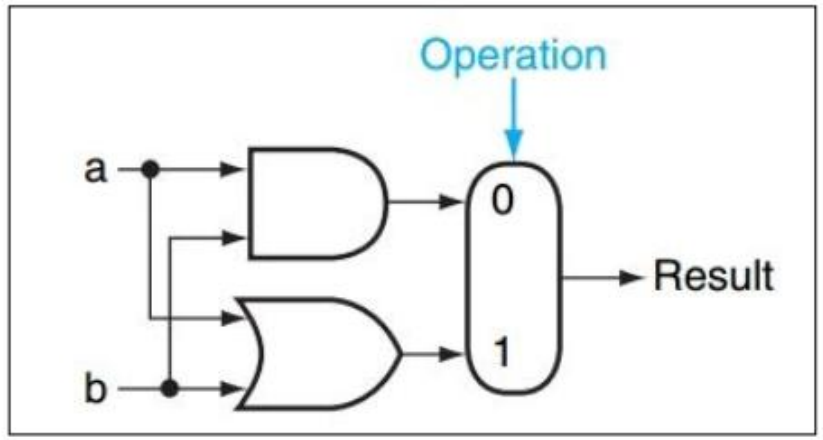
- المسألة: اختيار مصدر القيمة التي ستظهر على ممر
  - عرض الممر 32 bit
  - عدد المصادر الممكنة: 2
  - هذا يعني: عدد خطوط الاختيار اللازمة هو 1
  - نحتاج إلى 32 ناخبا بسيطا تشكل "مصفوفة" Array

Arithmetic and Logic Unit (ALU) Design

- خطوات التصميم
  - يبدأ التصميم بوحدة حساب ومنطق من أجل 1bit تستطيع إنجاز
    - AND
    - OR
  - إضافة إمكانية الجمع (مع حمل)
  - بناء ALU بعرض 32bit
  - إضافة إمكانية الطرح
  - إضافة إمكانية إجراء عملية NOR
  - إضافة العمليات الخاصة بالمعالج قيد التصميم
    - في حالة MIPS يجب إضافة إمكانية
      - إجراء عملية Set on less than slt
      - عمليات القفز المشروط beq, bne

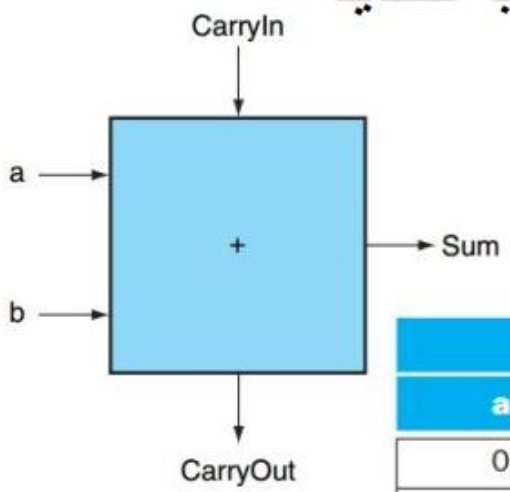
1-Bit ALU for AND and OR

- من خلال خط التحكم بالناخب يتم تحديد العملية التي سوف تظهر نتيجتها على الخرج



الجامع الكامل

- لإضافة إمكانية الجمع إلى وحدة الحساب والمنطق يجب تحديد كيفية تحقيق جامع كامل
  - الجامع الكامل: له ثلاثة مداخل ومخرجان
  - الجدول المنطقي للجامع الكامل



Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$

- ملاحظة: نصف الجامع له مدخلان فقط ومخرجان
  - هل يكفي نصف الجامع لتصميم الوحدة المطلوبة؟

تحقيق الجامع الكامل

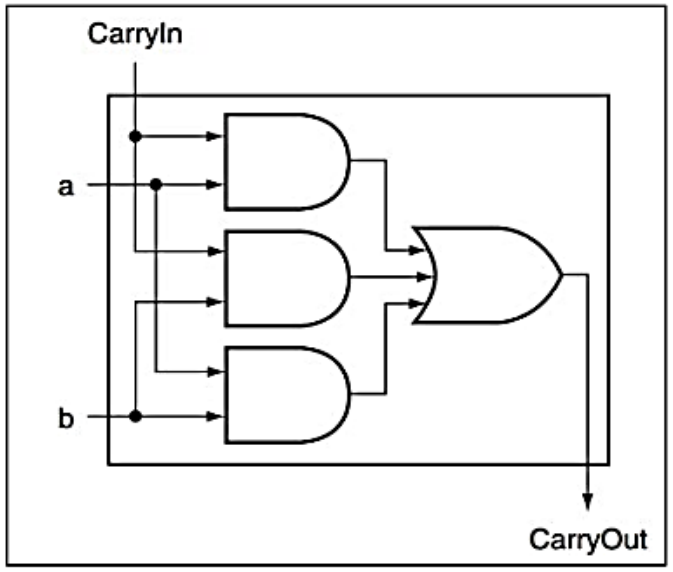
- يمكن كتابة التعبير المنطقي عن الـ CarryOut كما يلي

$$CarryOut = (b \cdot CarryIn) + (a \cdot CarryIn) + (a \cdot b) + (a \cdot b \cdot CarryIn)$$

- ويختصر إلى

$$CarryOut = (b \cdot CarryIn) + (a \cdot CarryIn) + (a \cdot b)$$

- ويحقق بالشكل

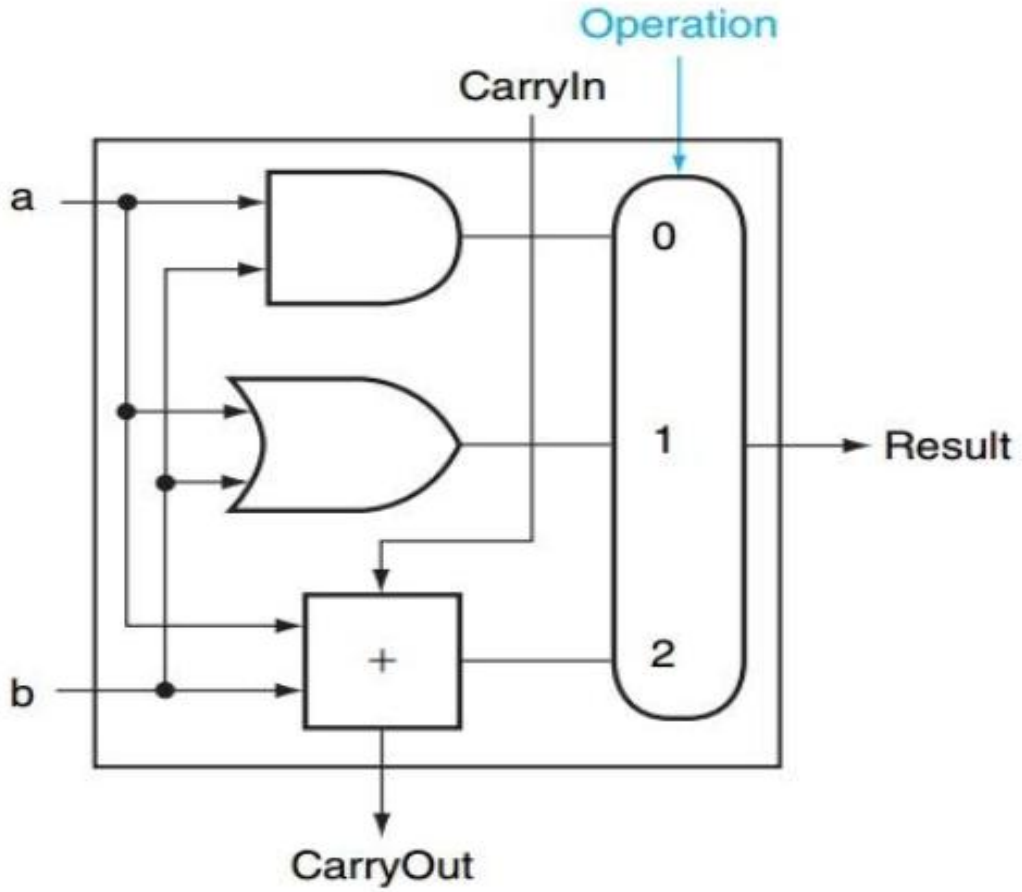


- الخرج Sum له التعبير المنطقي

$$Sum = (a \cdot \overline{b} \cdot \overline{CarryIn}) + (\overline{a} \cdot b \cdot \overline{CarryIn}) + (\overline{a} \cdot \overline{b} \cdot CarryIn) + (a \cdot b \cdot CarryIn)$$

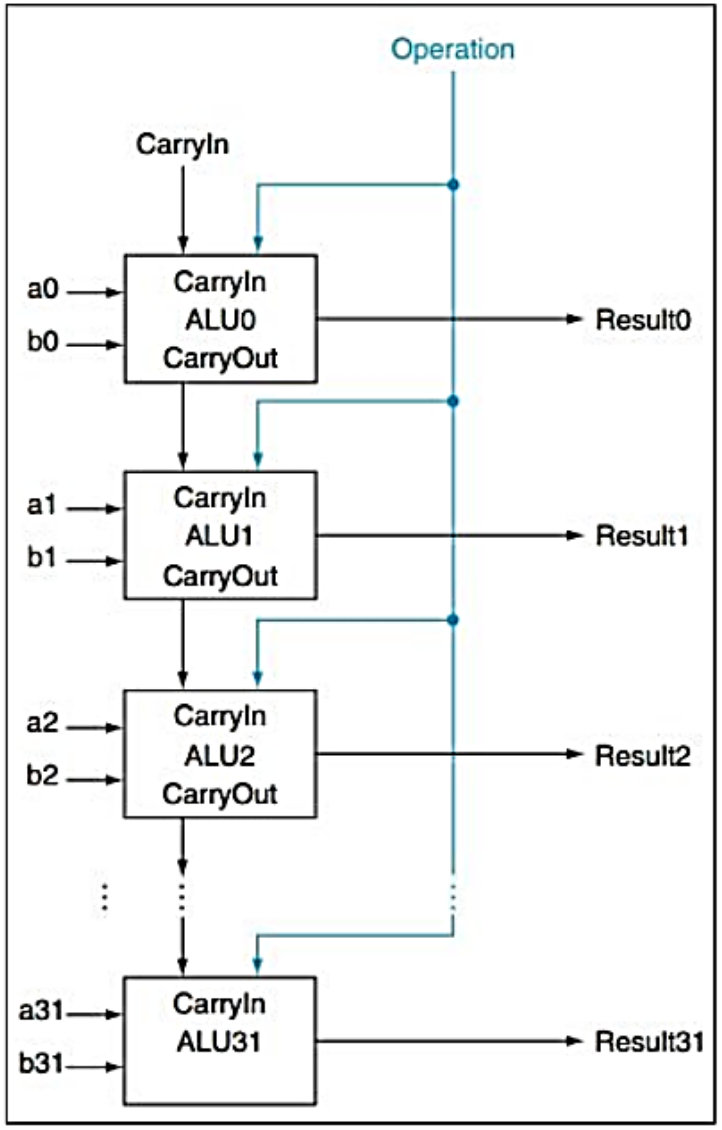
إضافة الجامع الكامل إلى الـ 1-bit ALU

Arithmetic and Logic Unit (ALU) Design



بناء 32-ALU

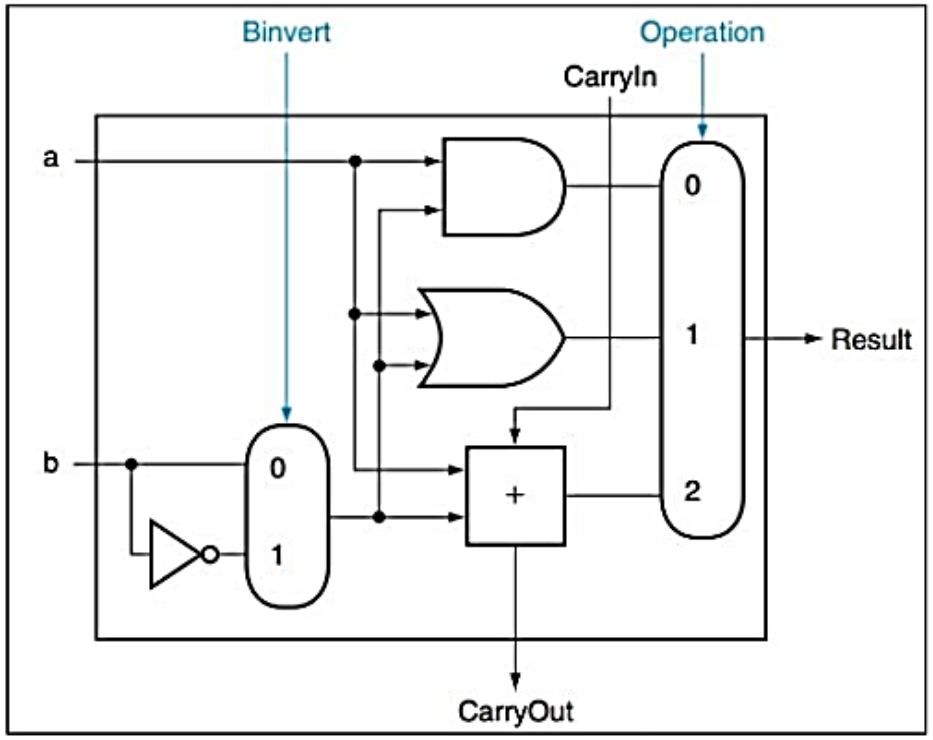
Arithmetic and Logic Unit (ALU) Design



- كل CarryOut يرتبط بالـ CarryIn لـ 1-Bit ALU اللاحقة
- يسمى الجامع الناتج بـ Ripple Carry Adder

إضافة إمكانية الطرح

- الطرح يعادل جمع القيمة السالبة للعدد
  - القيمة السالبة: أي المتمم الثنائي
  - للحصول على المتمم الثنائي يجب
    - عكس البتات
    - إضافة 1
- تحقيق الطرح
  - إضافة إمكانية عكس أحد المداخل
    - المدخل b



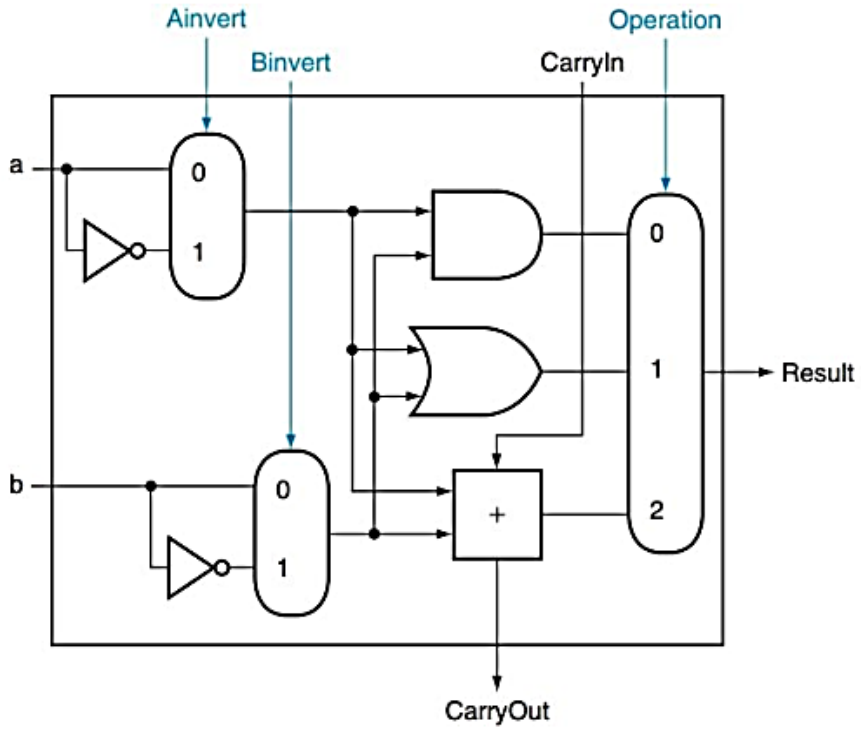
- إضافة 1 عن طريق وضع 1 فى CarryIn للبت الأقل أهمية

$$a + \overline{b} + 1 = a + (\overline{b} + 1) = a + (-b) = a - b$$

- سيكون خط CarryIn هذا أقرب إلى خط تحكم!

إضافة إمكانية إجراء عملية NOR

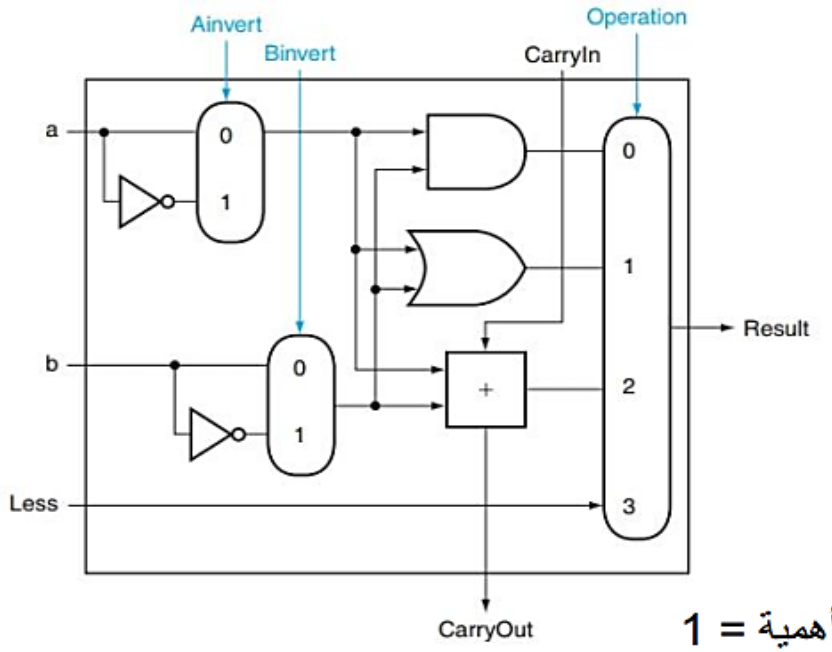
- يمكن إضافة عملية NOR كأية عملية ثنائية أخرى
  - بإضافة البوابة المنطقية اللازمة إلى كل 1-bit ALU
  - توسيع الناخب
  - توسيع خطوط التحكم



- لكن يمكن اللجوء إلى طرق أخرى
  - أقل كلفة
    - بالاستفادة من البوابات الموجودة
  - قانون DeMorgan: $\overline{(a+b)} = \bar{a} \cdot \bar{b}$
  - نحتاج فقط إلى عاكس على المدخل a

إضافة إمكانية إجراء slt

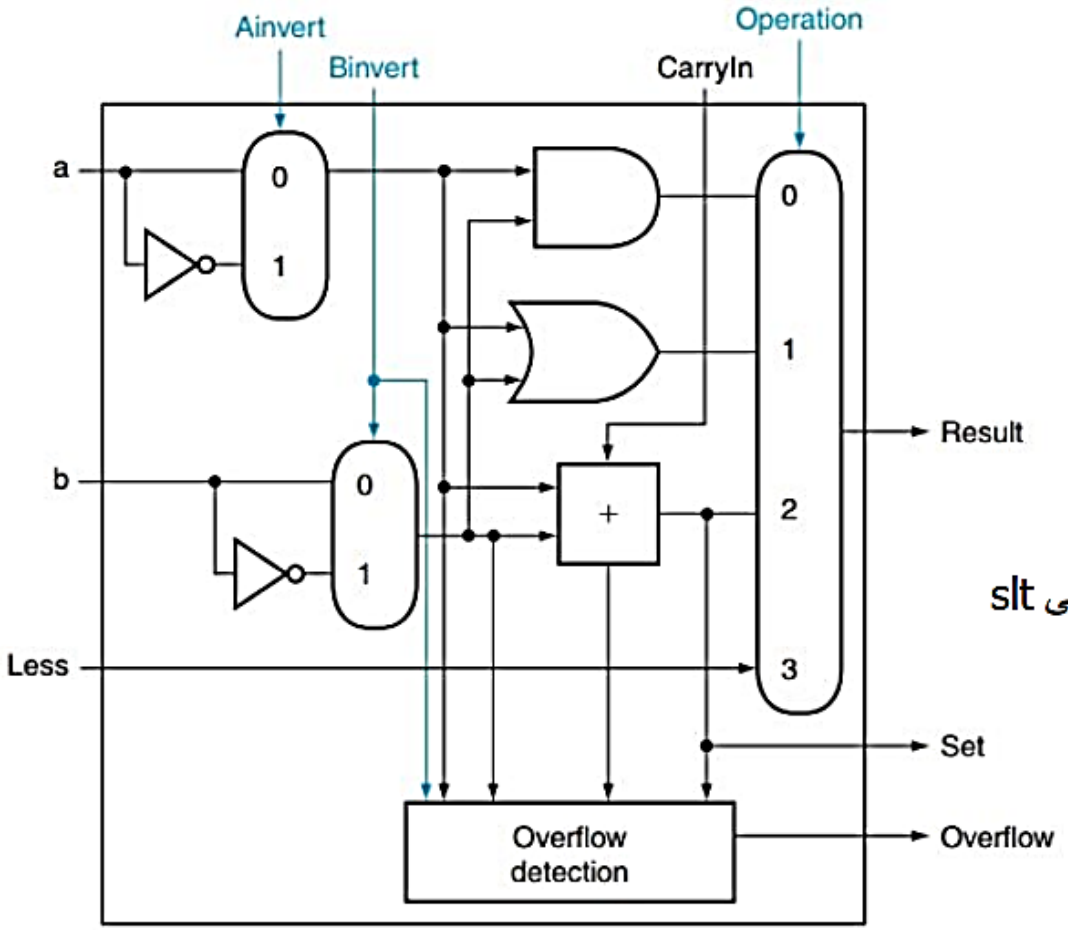
- ينتظر من العملية slt وضع 1 على الخرج عندما rs < rt



  - القيمة 1 تعني
    - البت الأقل أهمية على الخرج يساوي 1
    - بقية البتات أصفار
  - لتحقيق هذه الإمكانية
    - يضاف مدخل إلى الناخب خاص بهذه العملية
  - كيف ستتم المقارنة بين rs و rt؟
    - بإجراء طرح (rs – rt)
      - إذا كانت النتيجة سالبة يكون rs < rt
      - تكون النتيجة سالبة عندما يكون البت الأكثر أهمية = 1
    - التحقيق
      - توجه قيمة البت الأكثر أهمية في النتيجة نحو المدخل less للبت الأقل أهمية
      - تعطى بقية المداخل less دائما قيمة 0

إضافة إمكانية إجراء slt

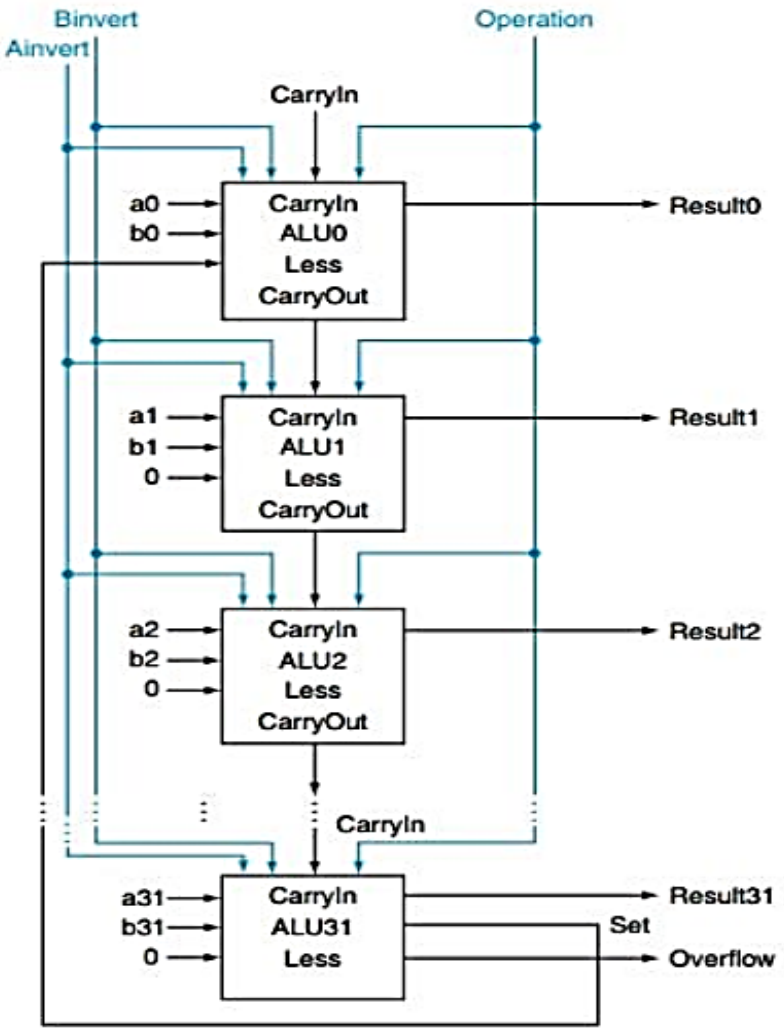
- للحصول على قيمة البت الأكثر أهمية الناتج
  - تصمم الـ 1-Bit ALU للبت الأكثر أهمية بشكل متميز عن غيرها
  - الخرج set هو قيمة البت الأكثر أهمية عند تنفيذ عملية الطرح



- ملاحظة
  - المخرج Overflow يؤثر حقيقة على slt

إضافة إمكانية إجراء slt

- طريقة الوصل لتحقيق slt



- الجزء الذي يتعلق بوحدة الحساب والمنطق هو اختبار تساوي أو عدم تساوي قيمتين
  - القفز سيتم أو لا يتم بناء على نتيجة الاختبار
  - تحقيق القفز لا يتعلق بوحدة الحساب والمنطق!
- كيف يمكن اختبار تساوي قيمتين
  - بالطرح

$$(a - b = 0) \Rightarrow a = b$$

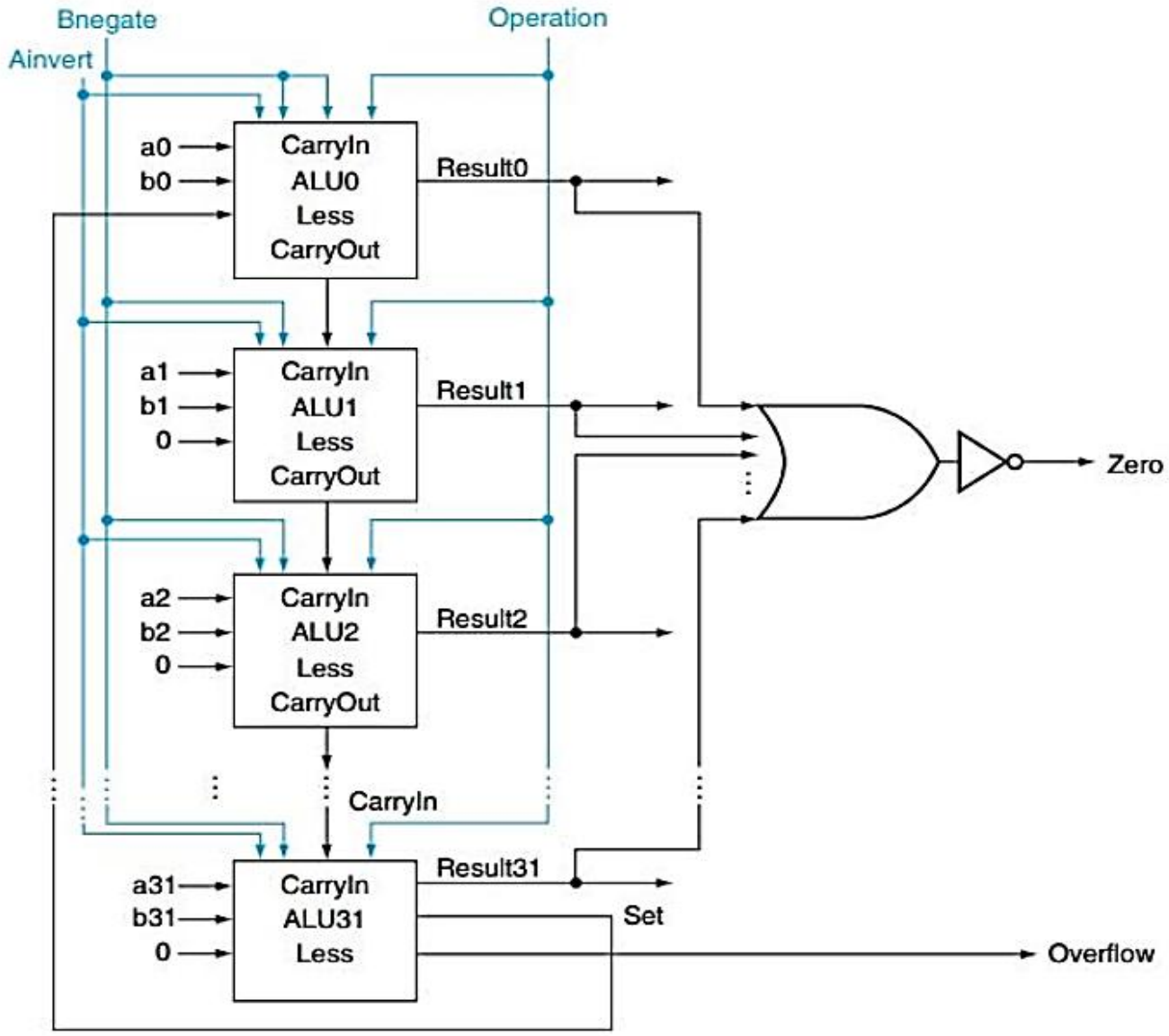
    - إمكانية متاحة في الـ ALU
  - واختبار كون النتيجة صفرية
    - الاختبار يحتاج إلى دارات خاصة لتحقيق:

$$\overline{Zero = (Result31 + Result30 + \ldots + Result2 + Result1 + Result0)}$$

إضافة إمكانية إجراء القفز المشروط

جَـامعة

Arithmetic and Logic Unit (ALU) Design



ماذا جرى لخطوط التحكم؟

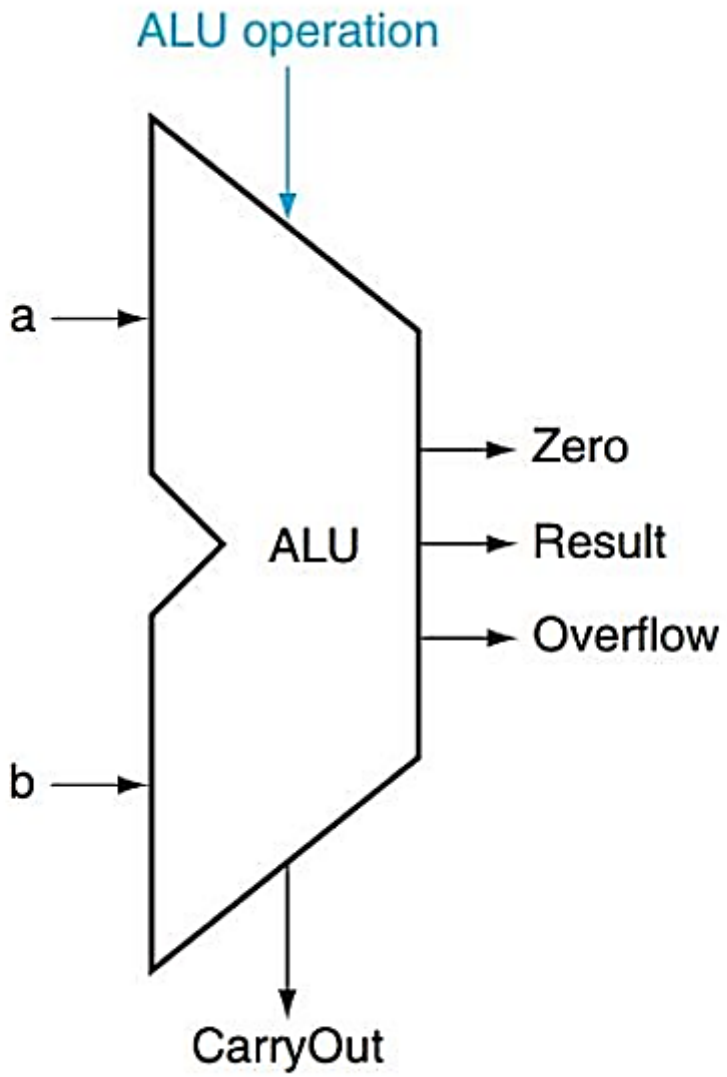
- الخطان Binvert و CarryIn يستعملان معا
  - يجب أن تكون قيمتهما مساوية لـ 1 لإجراء عملية الطرح
  - يمكن أن يستعمل Binvert لوحده من أجل عملية NOR
    - لاتؤثر قيمة CarryIn في هذه الحالة
  - لذلك يمكن توحيدهما في خط واحد اسمه Bnegate

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

- خطوط التحكم بالـ ALU
  - Ainvert
  - Bnegate
  - Operation (خطان للناخب مباشرة)
- يبين الجدول العملية التي تنفذها الـ ALU من أجل قيم خطوط التحكم

رمز وحدة الحساب والمنطق المصممة

Arithmetic and Logic Unit (ALU) Design



جَـامعة
المَـنارة
MANARA UNIVERSITY

نهاية المحاضرة الخامسة